ARMY RESEARCH LABORATORY

Radar Array Processing of Experimental Data Via the Scan-MUSIC Algorithm

by Canh Ly

ARL-TR-3135 **June 2004**

Army Research Laboratory
Adelphi, MD 20783-1197

ARL-TR-3135 **June 2004**

Radar Array Processing of Experimental Data Via the Scan-MUSIC Algorithm

Canh Ly
Sensors and Electron Devices Directorate, ARL

REPORT DOCUMENTATION PAGE			*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)* June 2004	**2. REPORT TYPE** Final	**3. DATES COVERED (From - To)** March 2003 to September 2003
4. TITLE AND SUBTITLE Radar Array Processing of Experimental Data Via the Scan-MUSIC Algorithm		**5a. CONTRACT NUMBER** **5b. GRANT NUMBER** **5c. PROGRAM ELEMENT NUMBER**
6. AUTHOR(S) Canh Ly		**5d. PROJECT NUMBER** **5e. TASK NUMBER** **5f. WORK UNIT NUMBER**
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) U.S. Army Research Laboratory Attn: AMSRD-ARL-SE-RM 2800 Powder Mill Road Adelphi, MD 20783-1197		**8. PERFORMING ORGANIZATION REPORT NUMBER** ARL-TR-3135
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) U.S. Army Research Laboratory 2800 Powder Mill Road Adelphi, MD 20783-1197		**10. SPONSOR/MONITOR'S ACRONYM(S)** **11. SPONSOR/MONITOR'S REPORT NUMBER(S)**

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This report demonstrates a novel algorithm of radar array processing that was developed at the U.S. Army Research Laboratory (ARL) and named the Scan-MUltiple SIgnal Classification (SMUSIC) algorithm. The algorithm can improve angular resolution of real targets when a single rotatable antenna is used as the sensor. The algorithm is an adaptation and extension of the MUSIC algorithm which requires an array of many sensors. We achieved a resolution of approximately 1/3 of antenna beamwidth for all target orientations and range offsets from exeprimental data. Previous work that used closely spaced point targets showed that the results from the SMUSIC algorithm could be degraded because of strong interference between the point target returns. Those interferences occur because of the relative phases of their returned signals. Therefore, there were some limitations of the algorithm for simple targets that consisted of a single scatterer. This new work shows that the limitation does not occur for real targets, which are more complex, such as would be present in Army radar applications.

We used a K_a-band, stepped frequency, pulsed radar to acquire the returns from two real targets, both located within a main lobe of the antenna pattern. The SMUSIC algorithm, which used magnitude-only experimental data, was then applied to accurately locate the centroids of the targets (an M60 tank and an M113 armored personnel carrier). All routines needed are written in MATLAB® and are included in appendix A.

MATLAB® is a registered trademark of The MathWorks.

15. SUBJECT TERMS

angular resolution, real scatterer-type targets, phase correction

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Canh Ly
a. REPORT UNCLASSIFIED	**b. ABSTRACT** UNCLASSIFIED	**c. THIS PAGE** UNCLASSIFIED	UL	60	**19b. TELEPHONE NUMBER** (*Include area code*) (301) 394-0868

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Tables

List of Figures

1 Introduction

The Rayleigh criterion states that the limit of angular resolution of an antenna aperture of size D is $\theta \approx \frac{\lambda}{D}$, in which θ is a multiple of $\frac{\lambda}{D}$, depending on the geometry of aperture, and λ is the wavelength corresponding to the transmitted frequency. The MUltiple SIgnal Classification (MUSIC) *(1)* algorithm, using the eigenvector decomposition method, is a super-resolution algorithm widely used to locate closely spaced multiple emitters (targets) with high resolution (better than the Rayleigh criterion). The algorithm is often employed with a uniform linear array of sensors of total aperture length D, and it relies on the phase progression between sensors of the returned signal. In contrast, a single antenna, also of aperture D and having a beam width θ, which is scanned in angle, generates no phase shifts within the main lobe because of the target direction; consequently, the MUSIC algorithm has not been used by others to super-resolve target positions within the main beam of a single narrow beam antenna. In this report, a novel technique of radar array processing that employs a scanning antenna to improve target detection in angular domain is described and assessed.

The algorithm, termed the Scan-MUSIC (MUltiple SIgnal Classification) or SMUSIC, was developed by the U.S. Army Research Laboratory (ARL) to improve the spatial resolution of two closely spaced point targets with a step-scanning radar antenna. These targets were physically located within the beam width of the antenna. However, there were some limitations of the algorithm for two, single point scatterer targets based on the experimental and computer simulation results shown in references *(2)* through *(5)*. The algorithm resolved the targets when they exhibited constructive interference; it could not resolve the targets when they exhibited destructive interference. In this report, the application of the algorithm was extended to a realistic complex scatterer-type target problem. The complex targets used were located within the beam width of a K_a-band radar antenna with frequency diversification, and the targets had different orientations and offsets.

This report also includes a graphical interface tool to run and analyze the algorithm for realistic complex scatterer-type targets. This tool, written in MATLAB, enables us to analyze the perfomance of the Scan-MUSIC algorithm in some practical scenarios and is a design tool to assist in design and development of future Army radar systems.

The report is divided into four sections. Section 2 describes experimental data collection. Section 3 presents the raw data, the SMUSIC results, and the tool to analyze experimental data. Section 4 draws a conclusion. In addition, all MATLAB®* codes of the tool are included in appendix A.

*MATLAB® is a registered trademark of The MathWorks.

2 Experimental Data Collection

Extensive field experimental data have been accumulated over the years with a linearly spaced sensor array. However, this has not been done for a scanning radar antenna. The objective of the experiment described in this report is to use a super-resolution algorithm, SMUSIC, for the analysis of the measured field data of the reflections of two real, complex scatterer-type targets via a scanning antenna. Subsection 2.1 of this section describes the experimental setup. Subsection 2.2 describes the data collection. Subsection 2.3 describes a procedure how to correct the phase information from the experimental data set.

2.1 Experimental Setup

The instrumentation radar system operates at K_a band (35 GHz) and was placed on the roof top of a truck. A circular antenna was mounted on a pedestal and had an aperture of 5 in. with conical narrow azimuth beam width. The radar was computer controlled and had a high data rate recording capability. Figure 1 shows the K_a-band radar mounted on the pedestal.

Figure 1: K_a-band radar and pedestal.

The experimental data for this SMUSIC analysis were collected at Aberdeen Proving Ground (APG), MD. The test range was a flat, grassy, open field. Two targets (armored personnel carrier [APC] M113 and M60 tank) were placed on the field at a range of approximately 600 m from the radar. These two targets were separated about 1/3 of the antenna beam width. Table 1 shows four test setups for this experiment. Figure 2 shows the first setup (Run 7) of the targets that were placed at about 600 m down-range position facing toward the radar. The range was measured to the approximate centroid of the targets. When we look at the figure, the left target is an M113 and the right one is an M60.

Run 8, shown in figure 3, kept the tanks at the same range as Run 7 but rotated the M60 tank at an angle approximately 40°.

Figure 2: Run 7, both tanks at the same down-range distance measured from center of the radar to the approximate target centroids.

Figure 3: Run 8, M60 tank rotated approximately 40°.

Table 1: Experimental test matrix for SMUSIC.

Run No.	Figure	Range @ 600 m	Target angle
7	2	Same	Head on
8	3	Same	M60 40° Offset
9	4	5 m closer	Head on
10	5	5 m closer	M60 40° Offset

Run 9, shown in figure 4, left the position of the M113 unchanged but moved the M60 tank to the radar about 5 m from the approximate centroid of the M113 to approximate centroid of the M60 tank.

Figure 4: Run 9, M60 tanks moved toward to the radar about 5 m measured from the two target centroids with no rotation.

Run 10, shown in figure 5, maintained the same target location as Run 9 and rotated the M60 tank at the same angle as in Run 8.

The antenna amplitude pattern was obtained with an aluminum trihedral radar cross section (RCS) = 19 dBsm. The reflector was placed 100 m down range on the radar line-of-sight axis. The average of the antenna magnitude data over the frequency band was used for the calculation with SMUSIC algorithm.

Figure 5: Run 10, same as in the third target setup with the M60 tank rotated approximately 40°.

2.2 Data Collection

The signals were collected as the antenna step-scanned horizontally across the target vehicles. At each step angle, the radar transmitted a coherent stepped frequency waveform which had 320 frequency steps from 32.4 GHz to 35.6 GHz with 10-MHz increments and at four different polarizations (vertical/vertical [VV], horizontal/horizontal [HH], vertical/horizontal [VH] and horizontal/vertical [HV]). The in-phase and quadrature-phase (I & Q) channel data were stored for five different range gates. Figure 6 shows the layout of the raw data stored as integer voltages (from analog-to-digital converter) in variable "out". In this figure, RGi stands for range gate i, where i = 1, ... 5; Az angles stand for azimuthal step angles. Frequency (320) is the index of stepped frequency.
The following is the MATLAB code used to format data as shown in figure 6:

```
data=reshape(out,[4 5 2 320 402]);
in which "data" is raw data format; "out" is a variable of an array
of 5,145,600 elements that will be reformatted (with a reshape MATLAB
function)in the format of 4 receiver channel (VI, VQ, HI, and HQ), 5 range
gates,2 polarizations (V or H), 320 step frequencies, and 402 step angles.
```

3 Results

This section describes the raw data format, the description of SMUSIC algorithm, and the performance of the algorithm and a tool to analyze the experimental data.

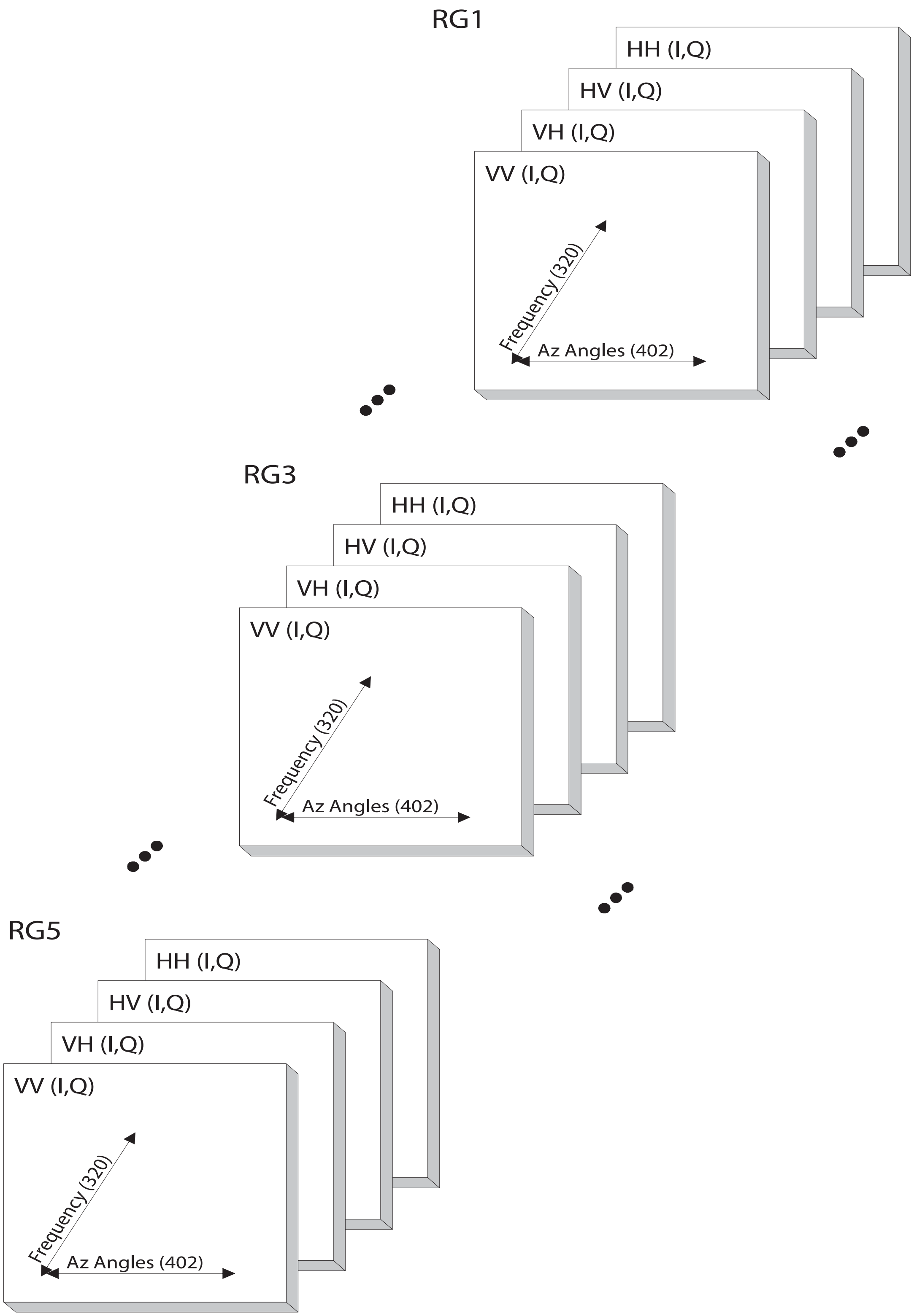

Figure 6: Raw data format.

3.1 Raw Data

The raw data were collected and stored in five range gates to verify that the targets were correctly placed in range gate 3. Since we know that the targets were placed at range gate 3, we use range gate 3 data and VV polarization for data processing. The motivation of this experiment was to demonstrate the use of frequency diversification to resolve targets in cross range, regardless of interferences. Since we have 320 frequencies, we will average all the frequencies to obtain the input data vector, $\boldsymbol{x}$, for the SMUSIC algorithm. The raw data format is shown in figure 7 for range gate 3.

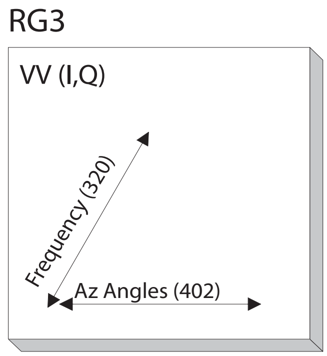

Figure 7: Range gate 3 raw data.

Figures 8 to 11 are magnitude data that were extracted from range gate 3 for each run corresponding to targets set up as shown in figures 2 through 5 in subsection 2.1. We use only the magnitude data in this case because the phase information from data does not add much value in this calculation because of the noisy and "uncorrected" phase from the experimental data. From those figures, we see that the returned signals do not show how many target centroids there are in the responses even though the angular separation between two targets was about 1/3 of the antenna beam width. Our goal is to use the SMUSIC algorithm to resolve target centroids from those data.

3.2 Scan-MUSIC Algorithm

Super-resolution algorithms, such as the MUSIC algorithm, have a special structure that uses the phase augmentation in elements of a linearly spaced sensor array. However, for a scanning antenna, the phase is almost constant within the main lobe of the transmitted antenna beam width, while the amplitude of the response varies. Therefore, the returned signal does not have the same structure as for an array of uniformly spaced sensors.

The algorithms that use the signal subspace method use the noise eigenvector subspace from a data correlation matrix. The correlation matrix is defined as $\boldsymbol{R} = E\left[\boldsymbol{x}\boldsymbol{x}^{\boldsymbol{H}}\right]$, in which $\boldsymbol{x}$ is the input data vector and E is the ensemble average from the snapshots. Since we have only one snapshot (one complete scan) for each experimental run, the SMUSIC algorithm fails to resolve closely spaced targets within the main lobe of the antenna beam width because

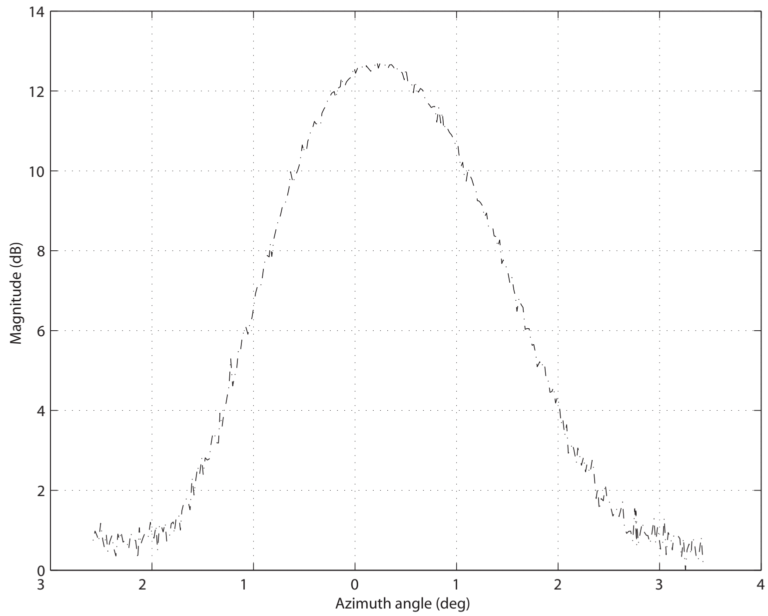

Figure 8: Experiment data, Run 7, range gate 3, polarization = VV.

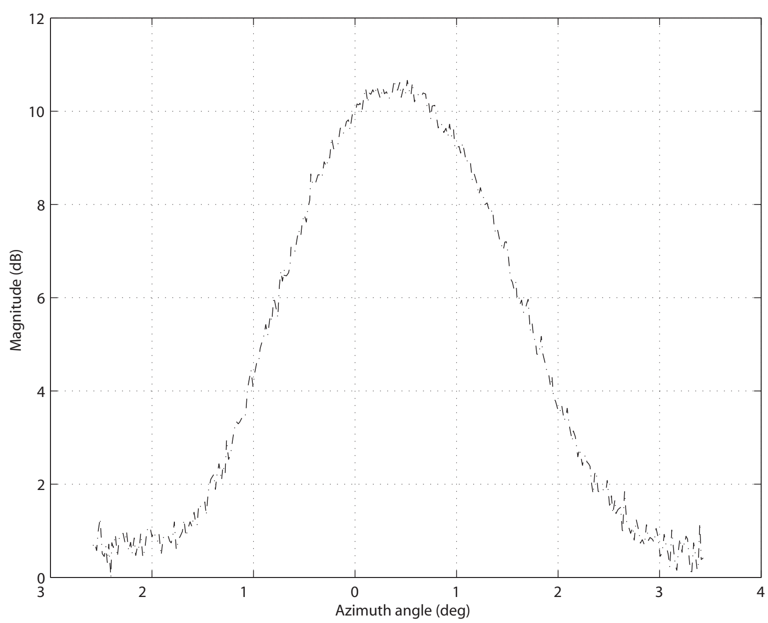

Figure 9: Experiment data, Run 8, range gate 3, polarization = VV.

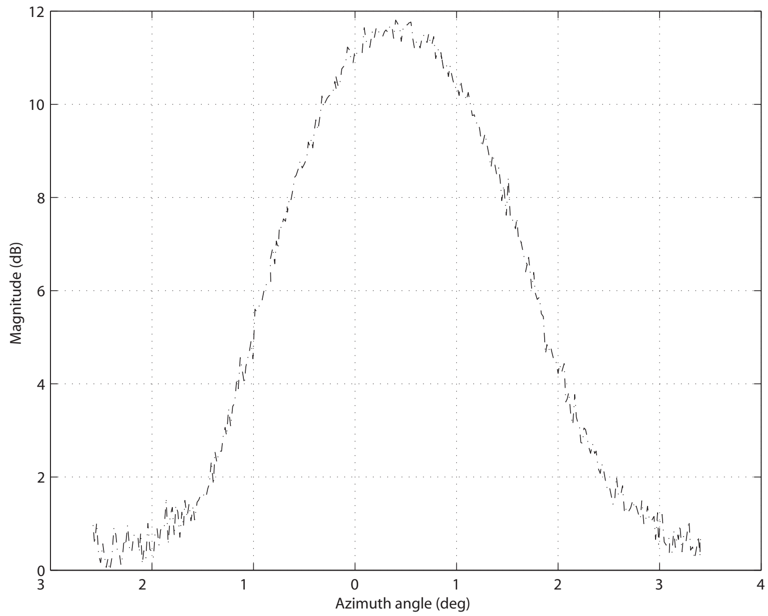

Figure 10: Experiment data, Run 9, range gate 3, polarization = VV.

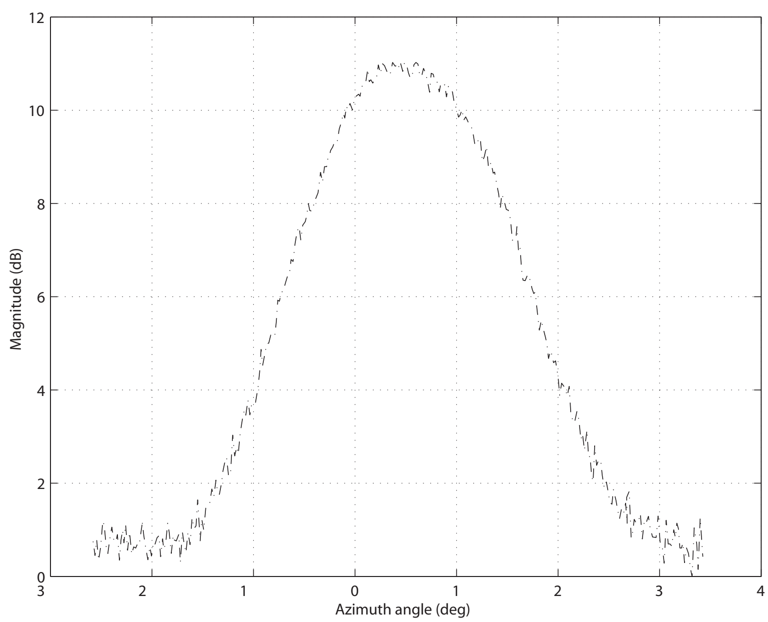

Figure 11: Experiment data, Run 10, range gate 3, polarization = VV.

of rank deficiency in the correlation matrix. To remedy this and to "decorrelate" coherent signals, Evans et al. *(9)* introduced the spatial smoothing method, and its proof was shown in Shan et al. *(10)* and Friedlander et al. *(11)*. This method was originally derived for the linearly spaced sensor array. We adapted the spatial smoothing method in the SMUSIC algorithm for a scanning radar antenna, called the subvector averaging method because the returned signal was received in a vector as the antenna stepped through discrete angles in the angular region of interest. This is a crucial step in producing a correlation matrix of sufficient rank as discussed more fully in reference *(2)*. Following is the description of the subvector averaging method.

Given input data $\boldsymbol{x}$ as described in section 3.1, let us partition the input data vector, $\boldsymbol{x} = [x_1, x_2, \ldots, x_N]^T$, with N step scans into overlapping subvectors of size Q, with elements $\{1, \ldots, Q\}$ forming the first subvector, with elements $\{2, \ldots, Q+1\}$ forming the second subvector, and so on, as follows:

$$\begin{aligned} \boldsymbol{x}^{\{1\}} &= [x_1, x_2, \ldots, x_Q]^T, \\ \boldsymbol{x}^{\{2\}} &= [x_2, x_3, \ldots, x_{Q+1}]^T, \\ &\vdots \\ \boldsymbol{x}^{\{P\}} &= [x_P, x_{P+1}, \ldots, x_N]^T, \end{aligned} \tag{1}$$

in which $P = N - Q + 1$ is the total number of subvectors, $P < Q$.

Next, let the instantaneous subcorrelation matrix $\hat{\boldsymbol{R}}^{\{p\}}$ be $\boldsymbol{x}^{\{p\}}\boldsymbol{x}^{\{p\}H}$. Finally, define the estimated correlation matrix $\hat{\boldsymbol{R}}_{\boldsymbol{xx}}$ to be the average of P subcorrelation matrices as

$$\hat{\boldsymbol{R}}_{\boldsymbol{xx}} = \frac{1}{P}\sum_{p=1}^{P} \hat{\boldsymbol{R}}^{\{p\}} = \frac{1}{P}\sum_{p=1}^{P} \boldsymbol{x}^{\{p\}}\boldsymbol{x}^{\{p\}H}. \tag{2}$$

The matrix $\hat{\boldsymbol{R}}_{\boldsymbol{xx}}$ is of the order $Q \times Q$. If there are two target centroids, there will be two predominant eigenvalues with the orders of magnitude larger than other eigenvalues from $\hat{\boldsymbol{R}}_{\boldsymbol{xx}}$. Let $\{\hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{E}}\}$ be the eigendecomposition of $\hat{\boldsymbol{R}}_{\boldsymbol{xx}}$, where the eigenvalues $\hat{\boldsymbol{\lambda}} = \left\{\hat{\lambda}_1, \hat{\lambda}_2, \ldots, \hat{\lambda}_Q\right\}$ are arranged in decreasing order of magnitude, with corresponding normalized eigenvectors $\hat{\boldsymbol{E}} = \{\hat{\boldsymbol{e}}_1, \hat{\boldsymbol{e}}_2, \ldots, \hat{\boldsymbol{e}}_Q\}$. Let us partition these into signal and noise subspace components with $\hat{\boldsymbol{E}}_{\boldsymbol{S}} = \{\hat{\boldsymbol{e}}_1, \hat{\boldsymbol{e}}_2, \ldots, \hat{\boldsymbol{e}}_M\}$ having the dimension $Q \times M$ and $\hat{\boldsymbol{E}}_{\boldsymbol{N}} = \{\hat{\boldsymbol{e}}_{M+1}, \hat{\boldsymbol{e}}_{M+2}, \ldots, \hat{\boldsymbol{e}}_Q\}$ having the dimension $Q \times (Q - M)$, so that $\hat{\boldsymbol{E}} = \{\hat{\boldsymbol{E}}_{\boldsymbol{S}}, \hat{\boldsymbol{E}}_{\boldsymbol{N}}\}$ and, similarly, $\hat{\boldsymbol{\lambda}}_{\boldsymbol{S}} = \left\{\hat{\lambda}_1, \hat{\lambda}_2, \ldots, \hat{\lambda}_M\right\}$ and $\hat{\boldsymbol{\lambda}}_{\boldsymbol{N}} = \left\{\hat{\lambda}_{M+1}, \hat{\lambda}_{M+2}, \ldots, \hat{\lambda}_Q\right\}$, so that $\hat{\boldsymbol{\lambda}} = \{\hat{\boldsymbol{\lambda}}_{\boldsymbol{S}}, \hat{\boldsymbol{\lambda}}_{\boldsymbol{N}}\}$. These eigenvector subspaces are orthogonal to each other. For this report, we know there are two targets, so we can split the signal subspace with dimension $Q \times 2$ and the noise subspace with dimension $Q \times (Q - 2)$. In a general case, one could use the minimum description length (MDL) technique *(12,13)* to estimate the number of targets before splitting the signal and noise subspaces.

Let us introduce the $\boldsymbol{a}(\beta)$ vector to be the effective array manifold vector for all allowable angles β in the *true* signal space. In this case, $\boldsymbol{a}(\beta)$ is the two-way magnitude pattern of the K_a-band radar. Note that the number of elements of the manifold vector must be the same number of elements in each column of the noise eigenvector subspace. Since the signal subspace contains information about the angles of arrival from each plane wave, the array manifold vectors from those angles are also orthogonal to the vectors in the noise subspace. The magnitude of the product of the estimated manifold vector and the noise subspace is therefore a small number, i.e., $\|\boldsymbol{a}^H(\beta)\hat{\boldsymbol{E}}_{\boldsymbol{N}}\|^2 = \epsilon$ or

$$\boldsymbol{a}(\beta)^H \hat{\boldsymbol{E}}_N \hat{\boldsymbol{E}}_N^H \boldsymbol{a}(\beta) = \epsilon, \tag{3}$$

in which ϵ is a small number.

Equation 3 is a null SMUSIC spectrum. If we normalize this equation, it becomes

$$\frac{\boldsymbol{a}(\beta)^H \hat{\boldsymbol{E}}_N \hat{\boldsymbol{E}}_N^H \boldsymbol{a}(\beta)}{\boldsymbol{a}(\beta)^H \boldsymbol{a}(\beta)} \ll 1. \tag{4}$$

The inverse of equation 4, i.e., the magnitude product between a manifold vector from all possible angles and the noise subspace matrix, called the SMUSIC pseudo-spectrum, is

$$\hat{\boldsymbol{P}}_{MU}(\beta) = \frac{\boldsymbol{a}(\beta)^H \boldsymbol{a}(\beta)}{\boldsymbol{a}(\beta)^H \hat{\boldsymbol{E}}_N \hat{\boldsymbol{E}}_N^H \boldsymbol{a}(\beta)}, \quad \text{for all } \beta. \tag{5}$$

The quantity of equation 5 is large for β so that $\boldsymbol{a}(\beta)$s are in the true signal space. We see that the peaks from this pseudo-spectrum are more *intuitively appealing.* $\hat{\boldsymbol{P}}_{MU}(\beta)$ is the profile of a one-dimensional pseudo-spectrum. The locations of the peaks of this spectrum are the estimated locations of the target centroids. Equation 5 only estimates the locations, not their strength. There is a separate task to determine the strength of the estimated signals by application of the least square solution technique *(1).* This report does not cover this task.

3.3 Numerical Implementation of the SMUSIC Algorithm

Equation 5 has the same form as an ordinary MUSIC calculation. The SMUSIC algorithm performs the search for all possible angles β in the searching angular region to locate the peaks of the SMUSIC pseudo-spectrum. This searching method, which goes through each computational loop as shown in figure 12 for each searching angle, consumes a lot of computational time. A new procedure or method has been developed, which takes advantage of the optimized MATLAB software, for a fast computation of the SMUSIC spectrum as shown in figure 13. The technique involves the following:

1. Pre-calculate $\boldsymbol{a}(\beta)$ at each angle for all the searched angles.
2. Store them in a file or a computer memory.
3. Create a searched array manifold matrix, $\boldsymbol{\mathcal{A}}(\beta)$ in which the columns of the matrix are the array manifold vector of the searched angles.

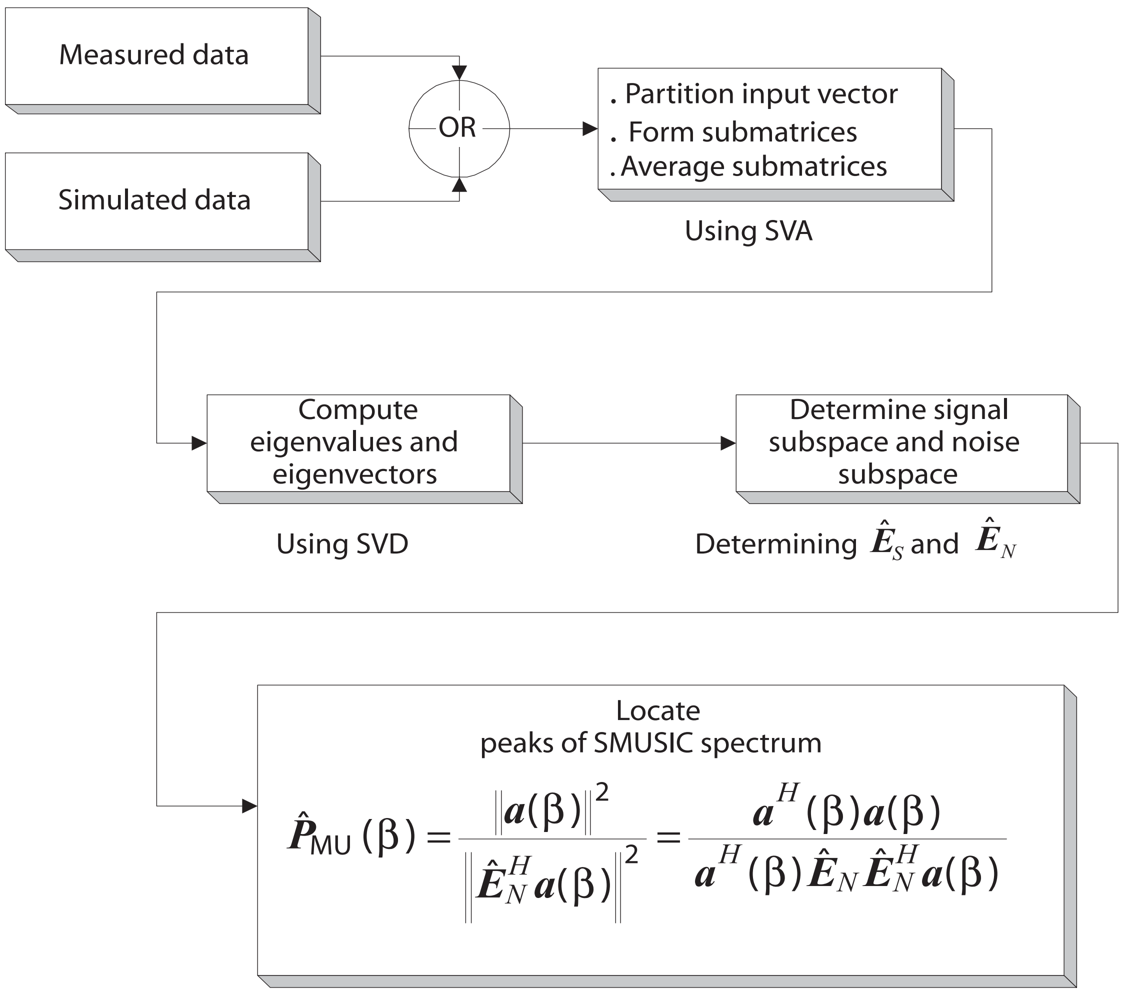

Figure 12: Ordinary SMUSIC process for experimental data and simulated data.

The SMUSIC pseudo-spectrum is shown below by the use of the property of the matrix product and the diagonal matrix manipulation to develop this fast technique:

$$\hat{\boldsymbol{P}}_{fast_{MU}}(\beta) = \frac{diag(\boldsymbol{\mathcal{A}}(\beta)^H \boldsymbol{\mathcal{A}}(\beta))}{diag(\boldsymbol{\mathcal{A}}(\beta)^H \hat{\boldsymbol{E}}_N \hat{\boldsymbol{E}}_N^H \boldsymbol{\mathcal{A}}(\beta))}, \quad \text{for all } \beta. \tag{6}$$

From equation 6, we see that the numerator is the vector of all the diagonal elements of the dot product of the searched array manifold matrix and its conjugate transpose. Since we know $\boldsymbol{\mathcal{A}}(\beta)$, the only thing that we need in the denominator is to compute the noise eigenvector subspace $\hat{\boldsymbol{E}}_{\boldsymbol{N}}$.

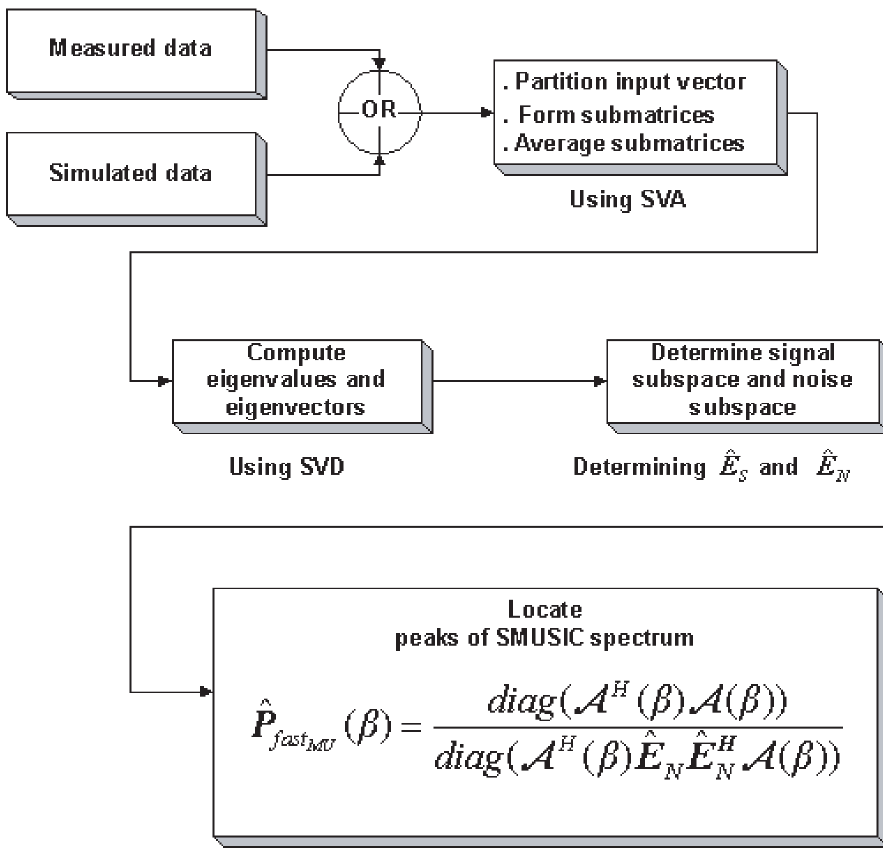

Figure 13: Fast SMUSIC process for experimental data and simulated data.

In summary, from those figures, SVA stands for subvector averaging method, and SVD stands for singular value decomposition. By comparing the ordinary MUSIC as shown in figure 12 and its fast implementation in figure 13, we see that the equation in the last block of the figure 12 is replaced with equation 6 for the fast implementation as shown in figure 13. We used the fast implementation for the analysis in the simulation. The fast implementation is about 100 times faster than the ordinary implementation.

3.4 Tool to Analyze Experimental Data

In this section, we describe an interface program. The interface program was written in the MATLAB language, version 6 or R12. The MATLAB code for this program is listed in appendix A. To run this program, first you need to invoke the MATLAB command window. Then, assumming that the interface program is in the current directory, type `smusicinter` Ver3 at the MATLAB prompt sign $\gg$ in the MATLAB command window. The SMUSIC interface window will appear on the screen. Figure 14 shows the interface program window which consists of the plotting window and the input parameter section. The input parameter section includes three parts: input file, processing, and plotting options.

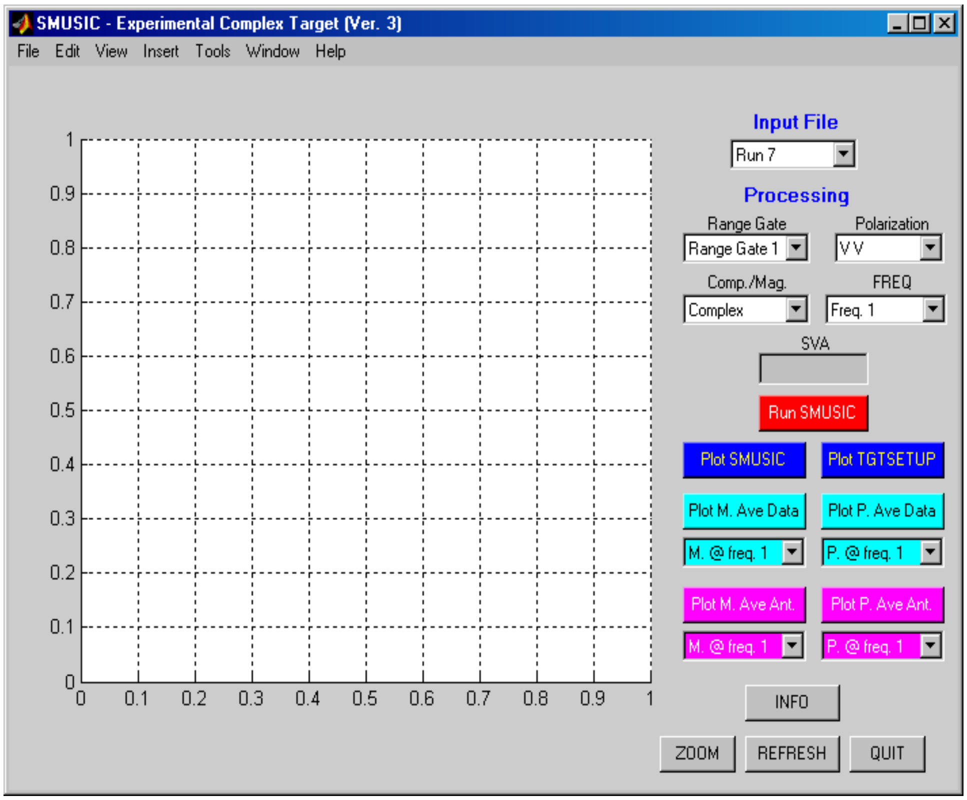

Figure 14: Interface program window.

3.4.1 Input File Selection

This selection allows users to select different experimental data. The data files used are Run7, Run8, Run9, and Run10, as described section 2.

3.4.2 Processing

This section allows users to select

- Range Gate – five range gates (1 to 5).
- Polarization – VV, VH, HV, HH.
- Comp./Mag. – compute SMUSIC pseudo-spectrum using either complex or magnitude data.
- FREQ – select a particular frequency or the average of all frequencies for the SMUSIC calculation.
- SVA – Number of subvector averages, i.e., the number of times that we perform subvector averaging to estimate the data correlation matrix. For example, SVA = 10 means that we perform 10 partitions of the input data, form these subcorrelation matrices, and then calculate the average of those subcorrelation matrices.
- Run SMUSIC – the execution button starts the SMUSIC calculation.

3.4.3 Plotting and Other Options

This section allows users to select

- Plot SMUSIC – plots SMUSIC pseudo-spectrum for the simulation after the SMUSIC calculation is done.
- Plot TGTSETUP – plots experimental target setup, depending on the input file selection.
- Plot M. Ave. Data – plots average magnitude of experimental data. This calculation is the average of all frequency steps within the radar beam width.
- Plot P. Ave. Data – plots average phase of experimental data. This calculation is the average of all frequency steps within the radar beam width.
- M. @ freq 1 ... – plots magnitude raw data at a paticular frequency which is used for the SMUSIC calculation.
- P. @ freq 1 ... – plots phase raw data at a particular frequency which is used for the SMUSIC calculation.
- Plot M. Ave. Ant. – plots average magnitude of the measured antenna pattern. This calculation is the average of all frequency steps within the radar beam width.

- Plot P. Ave. Ant. – plots average phase of the measured antenna pattern. This calculation is the average of all frequency steps within the radar beam width.
- M. @ freq 1 ... – plots magnitude antenna at a particular frequency which is used for the SMUSIC calculation.
- P. @ freq 1 ... – plots phase at a particular frequency which is used for the SMUSIC calculation.
- Info – This option shows where the MATLAB source code of the interface program is located.
- ZOOM – is an option to allow users to span the figure at a specific area. To do this, first click the ZOOM button, and then move the mouse cursor in the desired area of the figure and click the left mouse button. If you want to zoom back to the normal size figure, click the right mouse button.
- REFRESH – clears all input parameters, figure, and screen command window.
- QUIT – exits the interface program.

3.5 SMUSIC and Experimental Data Results

Figures 15 through 18 show the windows of the interface program and the results of the SMUSIC algorithm with selected parameters for the simulation. We selected experimental data at range gate 3, VV polarization, average magnitude data. These figures show the performance of the SMUSIC algorithm and the experimental data as a function of azimuth angles. In each figure, the solid curve is the SMUSIC result and the dash-dot-dash curve is the experimental data. The shaded area is the −3-dB beam width of the antenna. The "peaks" of the solid curve are the estimation of the target centroid locations. We see clearly that we achieve our goal to resolve the targets that are well within the antenna beam width. In fact, the separation of the two peaks is about 1/3 of antenna beam width for four experimental runs.

In summary, the SMUSIC algorithm resolved the targets that were placed within the −3-dB beam width of the K_a-band radar antenna. Although we collected both real (in-phase) and imaginary (quadrature) experimental data, we used only the magnitude data that were the square root of the sum of square of the real and the imaginary to perform the SMUSIC calculation. The combination of the real and imaginary data formed complex experimental data.

4 Conclusions

The report has described the novel extension of the MUSIC algorithm to the Scan-MUSIC (SMUSIC) for complex scatterer-type targets. The extension showed clearly the processing scheme of the SMUSIC algorithm which used a single rotatable antenna without increasing

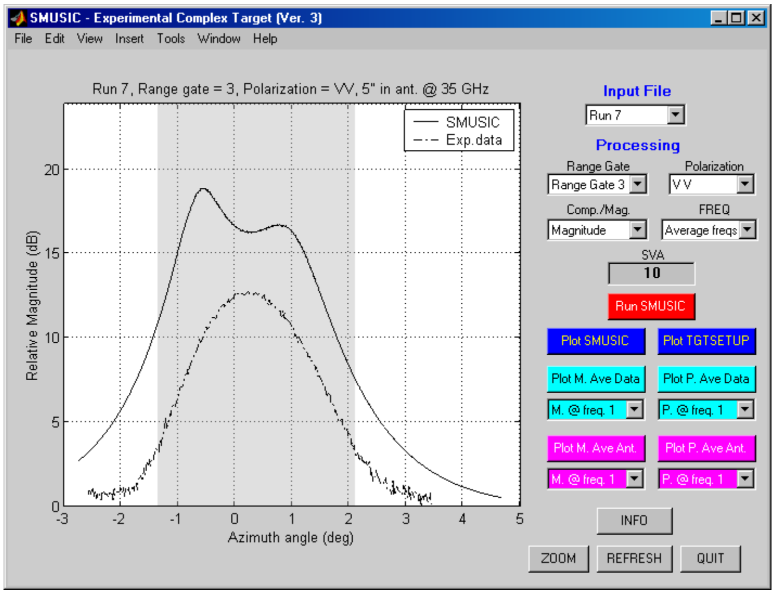

Figure 15: SMUSIC result and magnitude experimental data for Run 7, range 3, Polarization = VV, SVA = 10.

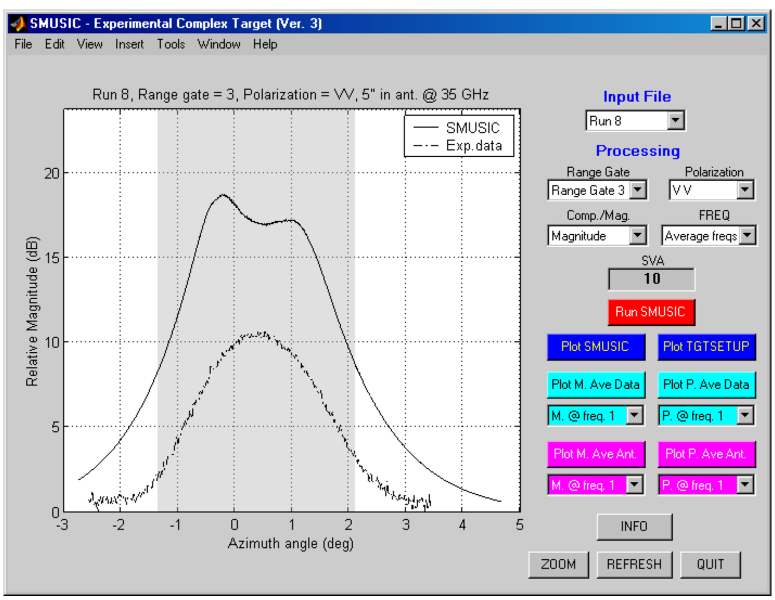

Figure 16: SMUSIC result and magnitude experimental data for Run 8, range 3, Polarization = VV, SVA = 10.

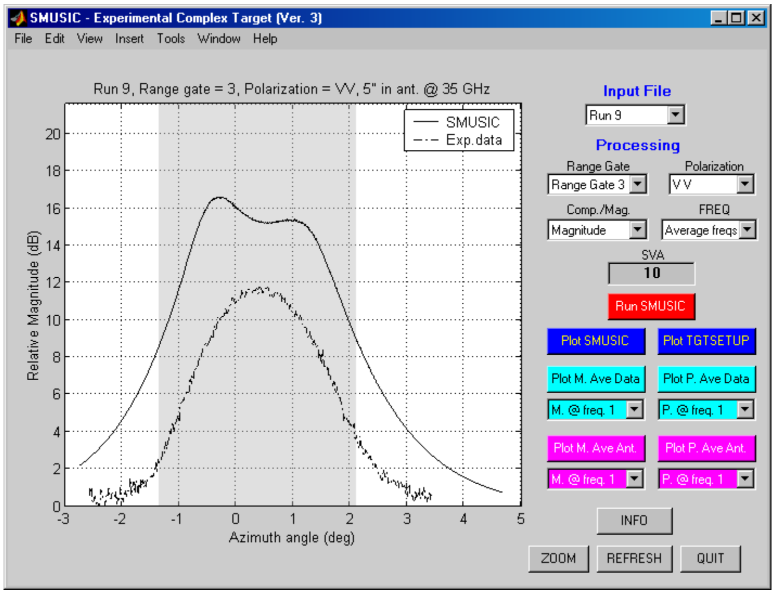

Figure 17: SMUSIC result and magnitude experimental data for Run 9, range 3, Polarization = VV, SVA = 10.

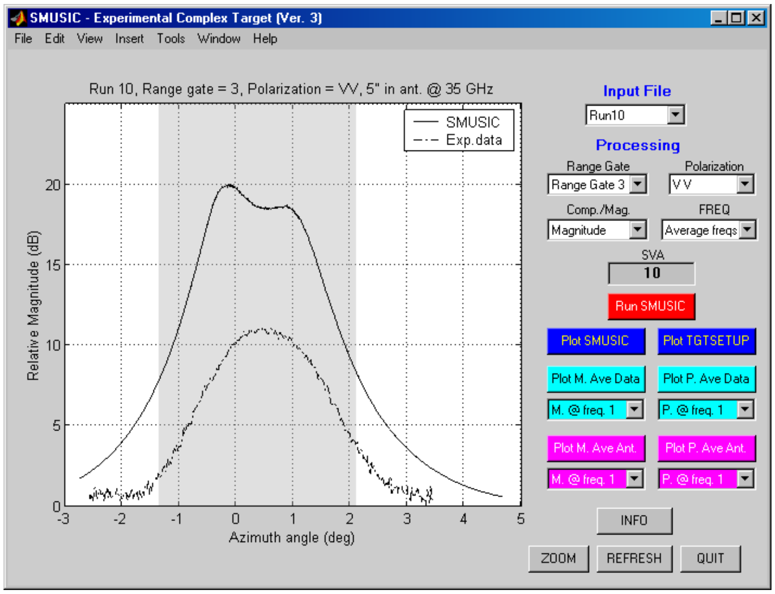

Figure 18: SMUSIC result and magnitude experimental data for Run 10, range 3, Polarization = VV, SVA = 10.

an antenna aperture. We used the SMUSIC algorithm with magnitude experimental data to resolve two closely spaced targets (an M60 tank and M113 APC) and to estimate the locations of their centroids in cross range with frequency diversification, regardless of orientation and offset of the two targets. We show that we can improve the angular resolution of the targets with SNR about 10 to 12 dBsm by about a factor of 3. We also described the graphical user interface program which allows us to analyze the performance of the SMUSIC algorithm. This report also described a fast implementation of the SMUSIC algorithm, which shows great promise in being implemented in a real-time radar system. The success of this implementation will enhance future U.S. Army radar systems.

References

[1] Schmidt, Ralph O. "Multiple Emitter Location and Signal Parameter Estimation." *IEEE Transactions on Antennas and Propagation*, March 1986, AP-34, No. 3, pp. 276-280.

[2] Ly, Canh. Angular Superresolution for Scanning Antenna, Ph.D. dissertation, George Mason Univeristy, Fairfax, Virginia, May 2000.

[3] Ly, Canh; Dropkin, Herbert; Manitius, Andre. "Array Processing Application: Angular Superresolution for Scanning Antenna." *Proceedings of the Thirty-Third Asilomar Conference on Signals, Systems, and Computers,* Pacific Grove, CA, October 1999, pp. 164-168.

[4] Dropkin, Herbert; Ly, Canh. "Superresolution for Scanning Antenna." *Proceedings of the 1997 IEEE National Radar Conference,* Syracuse, NY, May 1997, pp. 306-308.

[5] Ly, Canh; Dropkin, Herbert. *Superresolution for Low Cost Enabling Radar Technology (LCERT)*; ARL-TR-1780, U.S. Army Research Laboratory (ARL): Adelphi, Maryland, October 1998.

[6] Evans, J. E.; Johnson, J. R.; Sun, D. F. *Application of Advanced Signal Processing Techniques to Angle of Arrival Estimation in ATC Navigation and Surveillance*; Technical report 582; Lincoln Laboratory, M.I.T.: Lexington, Massachusetts, June 1982.

[7] Shan, Tie-Jun; Wax, Mati; Kailath, Thomas. "On Spatial Smoothing for Direction-of-Arrival Estimation of Coherent Signals." *IEEE Transactions on Acoustics, Speech, and Signal Processing*, August 1985, Vol. 33, No. 4, pp. 806-811.

[8] Friedlander, Benjamin; Weiss, Anthony J. "Direction Finding Using Spatial Smoothing with Interpolated Arrays." *IEEE Transactions on Aerospace and Electronic Systems*, April 1992; Vol. 28, No. 2, pp. 574-587.

[9] Wax, Mati; Kailath, Thomas. "Detection of Signals by Information Theoretic Criteria." *IEEE Transactions on Acoustics, Speech, and Signal Processing*, April 1985, Vol. 33, No. 2, pp. 387-392.

[10] Wax, Mati; Ziskind, Ilan. "Detection of the Number of Coherent Signals by the MDL Principle," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, August 1989, Vol. 37, No. 8, pp. 1190-1196.

A MATLAB source code of the Interface Program

```
% Filename: smuexpinter_Ver3.m (SMUSIC experimental interface file)
% Author: Canh Ly
% Date: 18 August 2003
%
% SMUSICEXPINTER_Ver2 plots experimental data and SMUSIC process using GUI.
% Example:
% Input File: smusic_run07
% Processing: Range Gate: Range Gate 3
% Polarization: VV
% Comp./Mag. : Complex
% One/Two Way: Two way
% SVA : 10
% Run SMUSIC
%
% Ver 2: 18 Aug 2003, Deleted One/Two Way Control Button
%                     Added FREQ to select indidual frequency or the
%                     average of all frequencies
% Ver 3: 9 February 2004: Delete Phase (Uncorrected and Corrected) button only used uncorrected

nsteps = 1;value=1; % Setup default number of steps and value for frequency selection
nfreqs = 1;
phase_value=1;
runno=7;range_gate=3;pol='VV';

figtag = findobj(0, 'tag', 'SMUSIC - Experimental Complex Target (Ver. 3)');

if isempty(figtag)

fig = figure('Name', 'SMUSIC - Experimental Complex Target (Ver. 3)','Units','pixels',
'Position', [310 250 700 495],...
'NumberTitle', 'off', 'resize', 'on',...
'tag', 'SMUSIC - Experimental Complex Target (Ver. 3)');

end
set(gcf, 'DefaultAxesPosition', [.075 .15 .6 .75]);

figcolor = get(gcf, 'color');

hndtexthead.Style = 'text';
hndtexthead.Units = 'normal';
hndtexthead.backgroundcolor = figcolor;
hndtexthead.foregroundcolor = 'blue';

hndtextsubhead.Style = 'text';
hndtextsubhead.Units = 'normal';
hndtextsubhead.backgroundcolor = figcolor;
hndtextsubhead.foregroundcolor = 'black';

% Input File Text position
```

```
uicontrol(gcf, hndtexthead,...
'position', [.755 .90 .15 .0385],...
'fontsize',10,'fontweight','bold',...
'string', 'Input File');

% Processing Text position
uicontrol(gcf, hndtexthead,...
'position', [.755 .80 .15 .0385],...
'fontsize',10,'fontweight','bold',...
'string', 'Processing');

uicontrol(gcf, hndtextsubhead,'Horiz','center',...
'position', [.72 .76 .115 .035],...
'string', 'Range Gate');

uicontrol(gcf, hndtextsubhead,'Horiz','center',...
'position', [.873 .76 .115 .035],...
'string', 'Polarization');

uicontrol(gcf, hndtextsubhead,'Horiz','center',...
'position', [.72 .68 .115 .035],...
'string', 'Comp./Mag.');

% uicontrol(gcf, hndtextsubhead,'Horiz','center',...
% 'position', [.873 .68 .115 .035],...
% 'string', 'One/Two Way');

uicontrol(gcf, hndtextsubhead,'Horiz','center',...
'position', [.873 .68 .115 .035],...
'string', 'FREQ');

% uicontrol(gcf, hndtextsubhead,'Horiz','center',...
% 'position', [.72 .595 .10 .035],...
% 'string', 'Phase');

% uicontrol(gcf, hndtextsubhead,'Horiz','center',...
% 'position', [.873 .595 .10 .035],...
% 'string', 'SVA');

uicontrol(gcf, hndtextsubhead,'Horiz','center',...
'position', [.80 .595 .10 .035],...
'string', 'SVA');


% Editable buttons
hndedit.Style = 'edit';
hndedit.Units = 'normal';
hndedit.Fontsize = 10;
hndedit.fontweight = 'bold';
% Define popup button
hndpopup.Style = 'popupmenu';
```

```
hndpopup.Units = 'normal';
% Define push button
hndpush.Style = 'push';
hndpush.Units = 'normal';

inputfilestr=['Run 7';
              'Run 8';
              'Run 9';
              'Run 10'];

inputfile= uicontrol(gcf, hndpopup, 'position', [.76 .85 .134 .05],...
'string', inputfilestr,'backgroundcolor','white','enable','on',...
'Interruptible','on','Horiz','center', ...
'foregroundcolor','black');
set(inputfile,'callback','setfile');

rangestr=['Range Gate 1|Range Gate 2|Range Gate 3|Range Gate 4|Range Gate 5'];
rangeinput = uicontrol(gcf, hndpopup, 'position', [.71 .72 .134 .05],...
'string', rangestr,'backgroundcolor','white','enable','on',...
'Interruptible','on', ...
'foregroundcolor','black');
set(rangeinput, 'callBack','setrangegate');

polstr=['V V|H H|V H|H V'];
polinput = uicontrol(gcf, hndpopup, 'position', [.87 .72 .115 .05],...
'string', polstr,'backgroundcolor','white','enable','on',...
'Interruptible','on', ...
'foregroundcolor','black');
set(polinput, 'callBack','setpol');


cmstr=['Complex|Magnitude'];
cminput = uicontrol(gcf, hndpopup, 'position', [.71 .635 .134 .05],...
'string', cmstr,'backgroundcolor','white','enable','on',...
'Interruptible','on', ...
'foregroundcolor','black');
set(cminput, 'callBack','setprocess');

% otstr=['Two way'];
% otinput= uicontrol(gcf, hndpopup, 'position', [.87 .635 .115 .05],...
% 'string', otstr,'backgroundcolor','white','enable','on',...
% 'Interruptible','on', ...
% 'foregroundcolor','black');
% set(otinput,'callback','setonetwo');

[data,ds] = extract21Apr03(runno,range_gate,pol);
nfreqs=size(ds.IQ_data,1);
labelstrm=[];
if nfreqs == 1
    labelstrm=['Freq. 1|'];
elseif nfreqs>=2
```

```
    temp = ['Freq. 1|'];
    for i = 2:nfreqs
        tempadd = ['Freq. ' num2str(i) '|'];
        labelstrm = [temp tempadd];
        temp = labelstrm;
    end
end
labelstrm=[labelstrm 'Average freqs'];
sfreqdata = uicontrol(gcf, hndpopup, 'position', [.87 .635 .115 .05],...
'string', labelstrm,'backgroundcolor','white','enable','on',...
'Interruptible','on', ...
'foregroundcolor','black');
freqsel = get(gco,'value');
set(sfreqdata,'callback', 'selectfreqdata');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 19 Mar 2003: Take out One way option out
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%otstr=['One way|Two way'];
%otinput= uicontrol(gcf, hndpopup, 'position', [.77 .46 .115 .05],...
%'string', otstr,'backgroundcolor','white','enable','on',...
%'Interruptible','on', ...
%'foregroundcolor','black');
%set(otinput,'callback','setonetwo');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% otstr=['Two way'];
% otinput= uicontrol(gcf, hndpopup, 'position', [.77 .46 .115 .05],...
% 'string', otstr,'backgroundcolor','white','enable','on',...
% 'Interruptible','on', ...
% 'foregroundcolor','black');
% set(otinput,'callback','setonetwo');

%
% phasestr=['Uncorrected|Corrected'];
% phaseinput = uicontrol(gcf, hndpopup, 'position', [.71 .555 .134 .05],...
% 'string', phasestr,'backgroundcolor','white','enable','on',...
% 'Interruptible','on', ...
% 'foregroundcolor','black');
% set(phaseinput, 'callBack','setphase');
%%%%%%%%%%%% 9 Feb 2004 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

phase_value=1;
if phase_value==1
    phastr='Uncorrected phase';
elseif phase_value==2
    phastr='Corrected phase';
end

% % Input SVA
% input1 = uicontrol(gcf, hndedit, 'position', [.87 .56 .115 .045]);
```

```
% call1 = ['sva = str2num(get(gco, ''string''));'];
% set(input1, 'callBack', call1);
%%%%%%%%%%%%% 9 February 2004 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Input SVA
input1 = uicontrol(gcf, hndedit, 'position', [.79 .56 .115 .045]);
call1 = ['sva = str2num(get(gco, ''string''));'];
set(input1, 'callBack', call1);

% Run SMUSIC
calcstart = uicontrol(gcf, hndpush, 'position', [.79 .495 .115 .05],...
'string', 'Run SMUSIC','backgroundcolor','red',...
'foregroundcolor','white','callback', 'smusic_mfrf_inter_Ver2');

% Plot SMUSIC and TGTSETUP
plotsmu = uicontrol(gcf, hndpush, 'position', [.71 .43 .13 .05],...
'string', 'Plot SMUSIC','backgroundcolor','blue',...
'foregroundcolor','yellow','callback', 'plotsmuresult');

plottgt = uicontrol(gcf, hndpush, 'position', [.855 .43 .13 .05],...
'string', 'Plot TGTSETUP','backgroundcolor','blue',...
'foregroundcolor','yellow','callback', 'plottgtsetup');

% Plot data
% First row, Left button
plotmdata = uicontrol(gcf, hndpush, 'position', [.71 .36 .13 .05],...
'string', 'Plot M. Ave Data','backgroundcolor','cyan',...
'foregroundcolor','black','callback', 'plotmaveexpdata');
% First row, right button
plotpdata = uicontrol(gcf, hndpush, 'position', [.855 .36 .13 .05],...
'string', 'Plot P. Ave Data','backgroundcolor','cyan',...
'foregroundcolor','black','callback', 'plotpaveexpdata');

% Second row, left button
% Initialized string for magnitude plot

labelstrm=[];
if nfreqs == 1
    labelstrm=['M. @ freq. 1|'];
elseif nfreqs>=2
    temp = ['M. @ freq. 1|'];
    for i = 2:nfreqs
        tempadd = ['M. @ freq. ' num2str(i) '|'];
        labelstrm = [temp tempadd];
        temp = labelstrm;
    end
end
labelstrm=[labelstrm 'Average M. data'];
plotmdatafreq = uicontrol(gcf, hndpopup, 'position', [.71 .30 .13 .05],...
'string', labelstrm,'backgroundcolor','cyan','enable','on',...
'Interruptible','on', ...
```

```
'foregroundcolor','black');
column = get(gco,'value');
set(plotmdatafreq,'callback', 'plotmfreqdata');


% Second row, right button
% Initialized string for phase plot
labelstrp = [];
if nfreqs == 1
    labelstrp=['P. @ freq. 1|'];
elseif nfreqs>=2
    temp = ['P. @ freq. 1|'];
    for i = 2:nfreqs
        tempadd = ['P. @ freq. ' num2str(i) '|'];
        labelstrp = [temp tempadd];
        temp = labelstrp;
    end
end
labelstrp=[labelstrp 'Average P. data'];
plotpdatafreq = uicontrol(gcf, hndpopup, 'position', [.855 .30 .13 .05],...
'string', labelstrp,'backgroundcolor','cyan','enable','on',...
'Interruptible','on', ...
'foregroundcolor','black');
column = get(gco,'value');
set(plotpdatafreq,'callback', 'plotpfreqdata');

% Plot antenna
% First row, left button
plotmaveant = uicontrol(gcf, hndpush, 'position', [.71 .23 .13 .05],...
'string', 'Plot M. Ave Ant.','backgroundcolor','magenta',...
'foregroundcolor','white','callback', 'plotantpat');

% First row, right button
plotpaveant = uicontrol(gcf, hndpush, 'position', [.855 .23 .13 .05],...
'string', 'Plot P. Ave Ant.','backgroundcolor','magenta',...
'foregroundcolor','white','callback', 'plotantpat');

% Second row, left button
% Initialized string for magnitude plot
labelstrm=[];
if nfreqs == 1
    labelstrm=['M. @ freq. 1|'];
elseif nfreqs>=2
    temp = ['M. @ freq. 1|'];
    for i = 2:nfreqs
        tempadd = ['M. @ freq. ' num2str(i) '|'];
        labelstrm = [temp tempadd];
        temp = labelstrm;
    end
end
labelstrm=[labelstrm 'Average M. Ant.'];
```

```
plotmantfreq = uicontrol(gcf, hndpopup, 'position', [.71 .17 .13 .05],...
'string', labelstrm,'backgroundcolor','magenta','enable','on',...
'Interruptible','on', ...
'foregroundcolor','white');
column = get(gco,'value');
set(plotmantfreq,'callback', 'plotmfreqant');

% Second row, right button
% Initialized string for magnitude plot
labelstrm=[];
if nfreqs == 1
    labelstrm=['P. @ freq. 1|'];
elseif nfreqs>=2
    temp = ['P. @ freq. 1|'];
    for i = 2:nfreqs
        tempadd = ['P. @ freq. ' num2str(i) '|'];
        labelstrm = [temp tempadd];
        temp = labelstrm;
    end
end
labelstrm=[labelstrm 'Average P. Ant.'];
plotpantfreq = uicontrol(gcf, hndpopup, 'position', [.855 .17 .13 .05],...
'string', labelstrm,'backgroundcolor','magenta','enable','on',...
'Interruptible','on', ...
'foregroundcolor','white');
column = get(gco,'value');
set(plotpantfreq,'callback', 'plotpfreqant');


% Push bottons
uicontrol(gcf, hndpush, 'position', [.775 .095 .10 .05],...
'string', 'INFO','callback', 'infofile');

uicontrol(gcf, hndpush, 'position', [.685 .025 .08 .05],...
'string', 'ZOOM','callback', 'zoom');

uicontrol(gcf, hndpush, 'position', [.775 .025 .10 .05],...
'string', 'REFRESH','callback', 'smuexpinterinit_Ver2');

uicontrol(gcf, hndpush, 'position', [.885 .025 .08 .05],...
'string', 'QUIT','callback', 'close all');
grid
clc;
```

```
% function setfile
% Filename: setfile.m
% Author  : Canh Ly
% Date    : 16 March 2001

file_value = get(gco,'value');
if file_value==1 % FFT calculation
    titlestr='Run 7';
    runno=7;
    A=imread('run7setup.jpg','jpg');
elseif file_value==2
    titlestr='Run 8';
    runno=8;
    A=imread('run8setup.jpg','jpg');
elseif file_value==3
    titlestr='Run 9';
    runno=9;
    A=imread('run9setup.jpg','jpg');
elseif file_value==4
    titlestr='Run 10';
    runno=10;
    A=imread('run10setup.jpg','jpg');
end

% function setrangegate
% Filename: setrangegate.m
% Author  : Canh Ly
% Date    : 10 December 2001

range_gate = get(gco,'value');


% function setpol
% Filename: setpol.m
% Author  : Canh Ly
% Date    : 10 December 2001

pol_value = get(gco,'value');
if pol_value==1 % FFT calculation
    pol='VV';
elseif pol_value==2
    pol='HH';
elseif pol_value==3
    pol='VH';
elseif pol_value==4
    pol='HV';
end


%function setprocess
% Filename: setprocess.m
```

```
% Author  : Canh Ly
% Date    : 16 January 2001

comp_mag = get(gco,'value');

%function selectfreqdata(freqsel)
% Filename: plotmfreqdata.m
% Author  : Canh Ly
% Date    : 18 August 2003
%
% Purpose: Select data at particular frequency for SMUSIC processing.
%

freqsel = get(gco,'value');

% function setphase
% Filename: setphase.m
% Author  : Canh Ly
% Date    : 14 August 2003

phase_value = get(gco,'value');
if phase_value==1
    phastr='Uncorrected phase';
elseif phase_value==2
    phastr='Corrected phase';
end
```

```
% Filename: smusic_mfrf_inter_Ver2.m
% Author: Canh Ly
% Date: 8 September 2003
% 1 April 2003: Modified the search algorithm as a fast search algorithm using the
%               property of the diagonal matrix operation

az=[42.0:0.2:44.8 45.0:0.01:55.0 55.02:0.2:57.0];
load amvmatout46 % This matrix file has a variable Vout, generated from
                 % genamvmat.m 25 April 2003

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Open antenna file and calculate phase correction
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[ant5,cal5] = antpat18Apr03(5,'VV',[1:320]); % All frequencies

%%%%%%%%%% Start from here: Old procedure of correcting phase%%%%%%%%%%%%%%
%
% tempant=ant5.IQ_data_raw.';% tempant: 320x791; ant5.IQ_data_raw: 791x320
% ffttempant=fft(tempant);
% magffttempant=abs(ffttempant);
% phaffttempant=angle(ffttempant);

% Calculate the phase at the maximum location of the average of all frequencies
% [Y,I]=max(ant5.yantdB_raw);
% fftantmax=fft(ant5.IQ_data_raw(I,:));
% [maxant,Imaxant]=max(abs(fftantmax));
% phasefftantmax=angle(fftantmax(Imaxant));

% [magffttempantmax,indexffttempantmax]=max(20*log10(magffttempant));
% angleffttempant=angle(ffttempant(indexffttempantmax));
% angleffttempoffset=angleffttempant-phasefftantmax;
% angleffttempoffset=angleffttempant-angleffttempant(1); % Correct all angle with
                                                         % repect to the first location
% A=angleffttempoffset'; % 791x1 dimension
% B=A(:,ones(1,320)); % 791x320 dimension
% C=B.';
% tempD=phaffttempant(:,232:682)-C(:,232:682);
% D=[phaffttempant(:,1:231) tempD phaffttempant(:,683:791)];
% E=D.';
%
% correctantmag=abs(ant5.IQ_data_raw);
% correctantpha=E;
% correctantpat=correctantmag.*exp(j.*E);
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%% End of old procedure of correcting phase %%%%%%%%

%%%%%%%%%% Start from here: new procedure of correcting phase %%%%%%%%%%%%%%
% 19 August 2003
%
ant5.IQ_data_raw_un=ant5.IQ_data_raw;
```

```
if phase_value==2
    phaseerror % Calculate phase error from phaseerror.m
    phaerrormat=phaerror(:,ones(1,size(ant5.IQ_data_raw,1)));
    antphaerror=unwrap(angle(ant5.IQ_data_raw.'))-phaerrormat;
    magIQ_data=abs(ant5.IQ_data_raw);
    phaIQ_data=antphaerror.';
    ant5.IQ_data_raw_cor=magIQ_data.*exp(j.*phaIQ_data);
    ant5.IQ_data_raw = ant5.IQ_data_raw_cor;
elseif phase_value==1
    ant5.IQ_data_raw = ant5.IQ_data_raw_un;
end
%%%%%%%%%%%%%%%%%%%%%%%%%% End of new procedure of correcting phase %%%%%%%

% Initialized string for antenna magnitude plot
nfreqs=size(ant5.IQ_data_raw,2);
labelstrm=[];
if nfreqs == 1
    labelstrm=['M. @ freq. 1|'];
elseif nfreqs>=2
    temp = ['M. @ freq. 1|'];
    for i = 2:nfreqs
        tempadd = ['M. @ freq. ' num2str(i) '|'];
        labelstrm = [temp tempadd];
        temp = labelstrm;
    end
end
labelstrm=[labelstrm 'Average M. Ant.'];
plotmantfreq = uicontrol(gcf, hndpopup, 'position', [.71 .17 .13 .05],...
'string', labelstrm,'backgroundcolor','magenta','enable','on',...
'Interruptible','on', ...
'foregroundcolor','white');
column = get(gco,'value');
set(plotmantfreq,'callback', 'plotmfreqant');

% Initialized string for antenna phase plot
abelstrm=[];
if nfreqs == 1
    labelstrm=['P. @ freq. 1|'];
elseif nfreqs>=2
    temp = ['P. @ freq. 1|'];
    for i = 2:nfreqs
        tempadd = ['P. @ freq. ' num2str(i) '|'];
        labelstrm = [temp tempadd];
        temp = labelstrm;
    end
end
labelstrm=[labelstrm 'Average P. Ant.'];
plotpantfreq = uicontrol(gcf, hndpopup, 'position', [.855 .17 .13 .05],...
'string', labelstrm,'backgroundcolor','magenta','enable','on',...
'Interruptible','on', ...
```

```
'foregroundcolor','white');
column = get(gco,'value');
set(plotpantfreq,'callback', 'plotpfreqant');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Open data file
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%[data,ds] = extract(runno,range_gate,pol);
[data,ds] = extract21Apr03(runno,range_gate,pol);
nfreqs=size(ds.IQ_data,1);

% Added on 18 Aug 2003 to correct phase of input data
ds.IQ_data_old=ds.IQ_data;
if phase_value==2
    phaseerror % Calculate phase error from calibration file
    pha_dataerrormat=phaerror(:,ones(1,size(ds.IQ_data,2)));
    phadsIQdata=unwrap(angle(ds.IQ_data))-pha_dataerrormat;
    magdsIQdata=abs(ds.IQ_data);
    ds.IQ_data = magdsIQdata.*exp(j.*phadsIQdata);
elseif phase_value==1
    ds.IQ_data = ds.IQ_data_old;
end

% Initialized string for data magnitude plot
labelstrm = [];
hndpopup.Style = 'popupmenu';
hndpopup.Units = 'normal';
if nfreqs == 1
    labelstrm=['M. @ freq. 1|'];
elseif nfreqs>=2
    temp = ['M. @ freq. 1|'];
    for i = 2:nfreqs
        tempadd = ['M. @ freq. ' num2str(i) '|'];
        labelstrm = [temp tempadd];
        temp = labelstrm;
    end
end
labelstrm=[labelstrm 'Average M. data'];
plotmdatafreq = uicontrol(gcf, hndpopup, 'position',[.71 .30 .13 .05],...
'string', labelstrm,'backgroundcolor','cyan','enable','on',...
'Interruptible','on', ...
'foregroundcolor','black');
column = get(gco,'value');
set(plotmdatafreq,'callback', 'plotmfreqdata');

% Initialized string for data phase plot
labelstrp = [];
if nfreqs == 1
    labelstrp=['P. @ freq. 1|'];
elseif nfreqs>=2
    temp = ['P. @ freq. 1|'];
```

```
    for i = 2:nfreqs
        tempadd = ['P. @ freq. ' num2str(i) '|'];
        labelstrp = [temp tempadd];
        temp = labelstrp;
    end
end
labelstrp=[labelstrp 'Average P. data'];
plotpdatafreq = uicontrol(gcf, hndpopup, 'position', [.855 .30 .13 .05],...
'string', labelstrp,'backgroundcolor','cyan','enable','on',...
'Interruptible','on', ...
'foregroundcolor','black');
column = get(gco,'value');
set(plotpdatafreq,'callback', 'plotpfreqdata');

if freqsel == size(ds.IQ_data,1)+1
    inputdata=[ds.outdatadB-min(ds.outdatadB)].';
else
    if comp_mag==1
       inputdata=ds.IQ_data(freqsel,:).';
    else
       inputdata=abs(ds.IQ_data(freqsel,:)).';
   end
end

Rxhat=estrxhatfs(inputdata,sva);

[eigval0,eigvec0]=eigvv(Rxhat);
[signal0,noise0]=snspaces(eigval0,eigvec0,2);
[nrownoisesum,ncolsnoisesum]=size(noise0);
esum0=noise0;
sumsuba_sum=0.0;
ncolA=size(Vout,2);
suba_sum = Vout(1:nrownoisesum,:);
num = diag(suba_sum'*suba_sum);
dem = diag(suba_sum'*(esum0*esum0')*suba_sum);
pmusics0=real(10*log10(num./dem));

plot(az,pmusics0,'b');hold
% set(gca,'Xlim',[40 58])
if (max(pmusics0)>= max(inputdata))
   axis([35 60 0 max(pmusics0)+5]);
else
   axis([35 60 0 max(inputdata)+5]);
end
plottitlestr=[titlestr,', Range gate = ',num2str(range_gate), ...
              ', Polarization = ',pol,', 5" in ant. @ 35 GHz'];
title(plottitlestr);
xlabel('Azimuth angle (deg)')
grid on
```

```
function [ant,cal] = antpat18Apr03(runno,pol,freqs)
% Filename: antpat18Apr03.m
% Author: Canh Ly
% Date: 18 April 2003
%
% Purpose: The program extracts data from smusic_run5.s16 files from the field
%          test on 25 October 2001.Then it plots the antenna pattern in which
%          the corner reflector was placed at 100 m (one range gate at 11) with
%          the RCS = 19.0 dBsm, or 79.4 m^2.
%
% Syntax:
%   [ant,cal] = antpat18Apr03(runno,pol,freqs);
%  where:
%   ant : data structure, output antenna pattern.
%
% Note:
%   Data for the antenna pattern is a multidimensional array. The dimension is
%   shown below:
%            - first dimension is receiver channel, 4 (VI, VQ, HI and HQ)
%            - second dimension is range gate, 1 (r=1)
%            - third dimension is polarization, 2 (p= V or H)
%            - fourth dimension is number of step frequencies, 320
%            - fifth dimension is azimuthal angles, 1002
%            The total size of the data file is = 4*1*2*320*1002 = 2565120
% Example:
%   1. Run antenna file number 5, VV polarization at freqs 190 to 290 or 34.4 to 35.4 GHz
%      [ant5,cal5] = antpat18Apr03(5,'VV',[190:290]);
%      [ant6,cal6] = antpat18Apr03(6,'VV',[190:290]);
%   2. Run antenna file number 5, VV polarization at freqs 1 to 320 or 33.2 to 35.4 GHz
%      [ant5,cal5] = antpat18Apr03(5,'VV',[1:320]);
%      [ant6,cal6] = antpat18Apr03(6,'VV',[1:320]);
%   3. Run antenna file number 5, VV polarization at freqs 100 to 320 or 34.2 to 35.4 GHz
%      [ant5,cal5] = antpat18Apr03(5,'VV',[100:320]);
%      [ant6,cal6] = antpat18Apr03(6,'VV',[100:320]);
%
if runno == 5
   sfilename = 'D:\cly\SMUSIC_MFRF\data\smusic_run5.s16';
   load  D:\cly\SMUSIC_MFRF\data\smusic_run5.tag;
   tagdata=smusic_run5; % Used to extract the second column for the azimuthal angles.
elseif runno==6
   sfilename = 'D:\cly\SMUSIC_MFRF\data\smusic_run6.s16';
   load  D:\cly\SMUSIC_MFRF\data\smusic_run6.tag;
   tagdata=smusic_run6; % Used to extract the second column for the azimuthal angles.
end

% Open calibration file
[cal] = calibration(5);

titlestr='Antenna Pattern';
if pol=='VV' | pol=='vv' | pol=='Vv' | pol== 'vV'
```

```
    receiver_pol = 1;
    pol_selection= 1;
elseif pol=='VH' | pol=='vh' | pol=='Vh' | pol== 'vH'
    receiver_pol = 3;
    pol_selection= 1;
elseif pol=='HH' | pol=='hh' | pol=='Hh' | pol== 'hH'
    receiver_pol = 3;
    pol_selection= 2;
elseif pol=='HV' | pol=='hv' | pol=='Hv' | pol== 'hV'
    receiver_pol = 1;
    pol_selection= 2;
end

titlestring=[titlestr ', polarization = ' pol];

% Read experimental data file and store the data in a multidimensional array.
fid = fopen(sfilename,'rb','b');
out = fread(fid,'int16');
data=reshape(out,[4 1 2 320 1002]); % See the above comments

% Extract offset data at the first azimuth angle, the first angle of 402 angles.
% Use 3.2 GHz bandwidth, corresponding from frequency 1 to frequency 320.
offset=data(:,:,:,freqs,1);
tfs = length(freqs); % Total number of frequency steps

% Calcualte average offset for VI channel
offset_VVI=offset(1,1,1,:);
offset_HVI=offset(1,1,2,:); % Added on 7 April 2003
% For all frequencies
avg_offset_VVI=sum(offset_VVI(:))/tfs;
avg_offset_HVI=sum(offset_HVI(:))/tfs;
ant.avg_offset_VVI = avg_offset_VVI;
ant.avg_offset_HVI = avg_offset_HVI; % Added on 7 April 2003

% Calculate average offset for VQ channel
offset_VVQ=offset(2,1,1,:);
offset_HVQ=offset(2,1,2,:); % Added on 7 April 2003
avg_offset_VVQ=sum(offset_VVQ(:))/tfs;
avg_offset_HVQ=sum(offset_HVQ(:))/tfs;
ant.avg_offset_VVQ = avg_offset_VVQ;
ant.avg_offset_HVQ = avg_offset_HVQ; % Added on 7 April 2003

% Calculate average offset for HI channel
offset_VHI=offset(3,1,1,:);
offset_HHI=offset(3,1,2,:); % Added on 7 April 2003
avg_offset_VHI=sum(offset_VHI(:))/tfs;
avg_offset_HHI=sum(offset_HHI(:))/tfs;
ant.avg_offset_VHI = avg_offset_VHI;
ant.avg_offset_HHI = avg_offset_HHI; % Added on 7 April 2003

% Calculate average offset for HQ channel
```

```
offset_VHQ=offset(4,1,1,:);
offset_HHQ=offset(4,1,2,:); % Added on 7 April 2003
avg_offset_VHQ=sum(offset_VHQ(:))/tfs;
avg_offset_HHQ=sum(offset_HHQ(:))/tfs;
ant.avg_offset_VHQ = avg_offset_VHQ;
ant.avg_offset_HHQ = avg_offset_HHQ; % Added on 7 April 2003
ant.avg_offset_VV = [avg_offset_VVI avg_offset_VVQ]; % Added on 7 April 2003
ant.avg_offset_VH = [avg_offset_VHI avg_offset_VHQ]; % Added on 7 April 2003
ant.avg_offset_HV = [avg_offset_HVI avg_offset_HVQ]; % Added on 7 April 2003
ant.avg_offset_HH = [avg_offset_HHI avg_offset_HHQ]; % Added on 7 April 2003
%
% Take out the second repeated data value in the signature file.
% First append one arbitrary number at the beginning of the file.
% Then find the difference of all data, if the same, the output is zero.
% Detect those zeros, and use only non-zero data.
%
theta = tagdata(2:size(tagdata,1),2);
nzeroindex=find((diff([1;theta]))~=0);
ant.theta_raw=theta(nzeroindex);
theta = theta(nzeroindex);

% Extract data
% Extract VI data for 320 frequencies and 401 azimuthal angles at range gate 3
% Use the full bandwidth
VIdata=reshape(data(receiver_pol,1,pol_selection,freqs,2:1002),tfs,1001);
% VIdata after taking out repeated data due to the pedestal jitter because
% of small anlge step size.
VIdata=VIdata(:,nzeroindex);
VQdata=reshape(data(receiver_pol+1,1,pol_selection,freqs,2:1002),tfs,1001);
% VQdata after taking out repeated data due to the pedestal jitter because
% of small anlge step size.
VQdata=VQdata(:,nzeroindex);

% Since there were two receivers, V and H each has I and Q ( or VI, VQ, HI and HQ),
% there were two independent transmitters V and H. Therefore, we can use VV (I & Q)
% and VH (I & Q) to calculate the offsets. Other channels such as HH (I & Q) and VH
% I & Q) are almost identical to others.

if pol=='VV' | pol=='vv' | pol=='Vv' | pol== 'vV'
   I_data=(VIdata-ant.avg_offset_VVI);
   Q_data=(VQdata-ant.avg_offset_VVQ);
elseif pol=='VH' | pol=='vh' | pol=='Vh' | pol== 'vH'
   I_data=(VIdata-ant.avg_offset_VHI);
   Q_data=(VQdata-ant.avg_offset_VHQ);
elseif pol=='HH' | pol=='hh' | pol=='Hh' | pol== 'hH'
   I_data=(VIdata-ant.avg_offset_VHI);
   Q_data=(VQdata-ant.avg_offset_VHQ);
elseif pol=='HV' | pol=='hv' | pol=='Hv' | pol== 'hV'
   I_data=(VIdata-ant.avg_offset_VVI);
   Q_data=(VQdata-ant.avg_offset_VVQ);
end
```

```
%%%%%%%
% Calculate the average of the magnitude antenna pattern for selected band of frequency.
%%%%%%%
% For all frequencies
outdata=sum(I_data.^2+Q_data.^2)/tfs;
ant.I_data_raw=I_data';
ant.Q_data_raw=Q_data';
ant.IQ_data_raw=ant.I_data_raw+j.*ant.Q_data_raw;
IQ_data=I_data+j.*Q_data;
ant.yant_raw=outdata.';
ant.yantdB_raw=(10*log10(outdata)).'; % Use 10*log10 because outdata is the
                                      % average of the sum of square

% 1. Compute the magnitude response of the antenna response at each frequency
% 2. Determine the maximum angle based on 1. at all frequencies (101)
% 3. Compute the average peak from 2.
% 4. Compute the error of the peak of antenna response at each frequency with
%    respect to the average peak
% 5. Calculate the standard deviation of the error of the peak of all antenna
%    responses.
magantmatrix=abs(ant.IQ_data_raw);
[maxmat,maxmatindex]=max(magantmatrix);
aveofpeak = sum(ant.theta_raw(maxmatindex))/tfs;
errorpeak=ant.theta_raw(maxmatindex)-aveofpeak;
stdpeak = std(errorpeak);

% 1. Locate the maximum antenna response
% 2. Normalize azimuthal scan angle to obtain "symmetrical" antenna
[maxmag,maxmagindex]=max(max(abs(ant.IQ_data_raw)));
%figure(1);plot(ant.theta_raw-(aveofpeak-stdpeak),abs(ant.IQ_data_raw(:,maxmagindex)));
outdatamag=abs(ant.IQ_data_raw(:,maxmagindex));
outdatacomplex=ant.IQ_data_raw(:,maxmagindex);
theta_freq81= ant.theta_raw-(aveofpeak-stdpeak)-0.0844;
[azscans81,magscans81,comp81]=antsmooth(theta_freq81,outdatamag,outdatacomplex,15);
skippts=3;
ant.azscan81 = azscans81(1:skippts:length(azscans81));
ant.comp81   = comp81(1:skippts:length(comp81));
ant.magscan81= (magscans81(1:skippts:length(magscans81)));
%-min(magscans81(1:skippts:length(magscans81)));

[Y,I]  = max(outdata);
yant=(outdata-Y).';
yantdB=(10*log10(outdata)).';
theta  = theta-theta(I);
ant.theta  = theta;
ant.yant=yant;
ant.yantdB=yantdB;

yant_test=(ant.yantdB_raw-min(ant.yantdB_raw));
yant_testmax=yant_test-max(yant_test);
% figure;plot(ant.theta,yant_testmax,'b')
```

```
if runno==5
   thetaindex=find(ant.yantdB-max(ant.yantdB)<=-2.92 & ant.yantdB-max(ant.yantdB)>=-3.07);
   angletwoindex=ant.theta(thetaindex);
   ant.beamwidth = angletwoindex(2)-angletwoindex(1);
end

lessm10index=find(theta<=-10.0);
gtm10index=find(theta>-10.0);
thetapatch=-(theta(lessm10index)+1);
thetaout = [theta(gtm10index); flipud(thetapatch)];
thetaindex=find(thetaout>=-10 & thetaout<=10);
xantout = thetaout(thetaindex);
ant.thetap = xantout;

yantpatch=yant(lessm10index);
yanttemp = [yant(gtm10index); flipud(yantpatch)];
yantout = yanttemp(thetaindex);

ant.yantp = yantout;
yantdBpatch=yantdB(lessm10index);
yantdBtemp = [yantdB(gtm10index); flipud(yantdBpatch)];
yantdBout = yantdBtemp(thetaindex);

ant.yantdBp = yantdBout;
I_datapatch=I_data(lessm10index);
I_datatemp= [I_data(gtm10index); flipud(I_datapatch)];
ant.I_data=I_datatemp(thetaindex);

Q_datapatch=Q_data(lessm10index);
Q_datatemp= [Q_data(gtm10index); flipud(Q_datapatch)];
ant.Q_data=Q_datatemp(thetaindex);

IQ_datapatch=IQ_data(lessm10index);
IQ_datatemp= [IQ_data(gtm10index); flipud(IQ_datapatch)];
ant.IQ_data=IQ_datatemp(thetaindex);

scanindex=find(ant.thetap>=-8.5 & ant.thetap<=8.5);
azscantemp=ant.thetap(scanindex);
magscantemp=ant.yantdBp(scanindex);
[azscans,magscans,magscans]=antsmooth(azscantemp,magscantemp,magscantemp,3);

skippts=2;
ant.azscan = azscans(1:skippts:length(azscans));
ant.magscan= (magscans(1:skippts:length(magscans)))-min(magscans(1:skippts:length(magscans)));
```

```
function [data,ds] = extract21Apr03(runno,range_gate,pol)
% Filename: extract21Apr03.m
% Author: Canh Ly
% Date: 21 April 2003
%
% Purpose: The program extracts data from *.s16 files from the field test on
%          25 October 2001.
%          Then it plots the magnitude response at the 'range_gate' and 'pol'
%          input selection.
%
% Syntax:
%   [data,ds] = extract21Apr03(runno,range_gate,pol);
%  where:
%   tagdata: input tagfile indicating how many azimuthal angles in each run.
%   data: output data is a multidimensional array. The dimension is shown below:
%         1. For data file (not antenna pattern)
%            - first dimension is receiver channel, 4 (c= I or Q), VI, VQ, HI and HQ
%            - second dimension is range gate, 5 ( r= 1, 2, 3, 4, 5)
%            - third dimension is polarization, 2 (p= V or H)
%            - fourth dimension is number of step frequencies, 320
%            - fifth dimension is azimuthal angles, 402
%            The total size of the data file is = 4*5*2*320*402 = 5145600
%         2. For antenna pattern file
%            - first dimension is receiver channel, 4 (c= I or Q), VI, VQ, HI and HQ
%            - second dimension is range gate, 1 (r=1)
%            - third dimension is polarization, 2 (p= V or H)
%            - fourth dimension is number of step frequencies, 320
%            - fifth dimension is azimuthal angles, 1002
%            The total size of the data file is = 4*1*2*320*1002 = 2565120
%   ds : data structure for the average of the offset values as the background
%   runno: run number from data signatures
%          [7] Run number 7 for SMUSIC_run7
%          [8] Run number 8 for SMUSIC_run8
%          [9] Run number 9 for SMUSIC_run9
%         [10] Run number 10 for SMUSIC_run10
%   range_gate: range gate selection (1 to 5)
%   pol       : polarization ('VV', 'HH', 'VH', or 'HV')
%
% Example:
%   [data,ds] = extract21Apr03(7,3,'VV');
%
% Modified: 8 April 2003: Added selection of frequency from 190 to 290 because
%           of flat phase response.
%

if runno == 7
    sfilename = 'D:\CLY\SMUSIC_MFRF\data\smusic_run7.s16';
    load  D:\CLY\SMUSIC_MFRF\data\smusic_run7.tag;
    tagdata=smusic_run7;
    titlestr='Run 7';
```

```
elseif runno == 8
    sfilename = 'D:\CLY\SMUSIC_MFRF\data\smusic_run8.s16';
    load  D:\CLY\SMUSIC_MFRF\data\smusic_run8.tag;
    tagdata=smusic_run8;
    titlestr='Run 8';
elseif runno == 9
    sfilename = 'D:\CLY\SMUSIC_MFRF\data\smusic_run9.s16';
    load  D:\CLY\SMUSIC_MFRF\data\smusic_run9.tag;
    tagdata=smusic_run9;
    titlestr='Run 9';
elseif runno == 10
    sfilename = 'D:\CLY\SMUSIC_MFRF\data\smusic_run10.s16';
    load  D:\CLY\SMUSIC_MFRF\data\smusic_run10.tag;
    tagdata=smusic_run10;
    titlestr='Run 10';
end

if pol=='VV' | pol=='vv' | pol=='Vv' | pol== 'vV'
    receiver_pol = 1;
    pol_selection= 1;
elseif pol=='VH' | pol=='vh' | pol=='Vh' | pol== 'vH'
    receiver_pol = 3;
    pol_selection= 1;
elseif pol=='HH' | pol=='hh' | pol=='Hh' | pol== 'hH'
    receiver_pol = 3;
    pol_selection= 2;
elseif pol=='HV' | pol=='hv' | pol=='Hv' | pol== 'hV'
    receiver_pol = 1;
    pol_selection= 2;
end

titlestring=[titlestr ', at range gate ' num2str(range_gate) ', and polarization = ' pol];

fid = fopen(sfilename,'rb','b');
out = fread(fid,'int16');

data=reshape(out,[4 5 2 320 402]); % See the above comments

% Extract offset data at the first azimuth angle, the first angle of 402 angles.
% Only use 2.2 GHz bandwidth, corresponding to from frequency 100 to frequency 320 (221 freqs).
% offset=data(:,:,:,100:320,1);
% Only use 1.0 GHz bandwidth, corresponding to from frequency 190 to frequency 290 (101 freqs).
% from 34.4 GHz to 35.4 GHz (8 April 2003)
% offset=data(:,:,:,190:290,1);
offset=data(:,:,:,1:320,1); % For all frequencies

% Modified: 31 Mar 2003
% For VI channel
% Compute the average offset of VI channel:
% 1. Obtain the offset of all range gates (1 to 5) for all frequencies (221 from
%    freq 100 to 320). The offset_VVI has a 5 x 221 dimension
```

```
% 2. Calculate the average offset of VI by averaging all frequencies (size(offset_VVI,2))
%    all columns) first and then for all range gates (size(offset_VVI,1)) (all rows).
% offset_VVI = reshape(offset(1,:,1,:),[5 221]);
% 8 April 2003: Choose frequencies from 190 to 290
% offset_VVI = reshape(offset(1,:,1,:),[5 101]);

offset_VVI = reshape(offset(1,:,1,:),[5 320]);
ds.avg_offset_VVI = sum(sum(offset_VVI,2)/size(offset_VVI,2))/(size(offset_VVI,1));

% For average offset of VQ channel
% offset_VVQ = reshape(offset(2,:,1,:),[5 221]); % For frequencies from 100 to 320
% 8 April 2003: Choose frequencies from 190 to 290
% offset_VVQ = reshape(offset(2,:,1,:),[5 101]);
offset_VVQ = reshape(offset(2,:,1,:),[5 320]);
ds.avg_offset_VVQ = sum(sum(offset_VVQ,2)/size(offset_VVQ,2))/(size(offset_VVQ,1));
% For HI channel

% offset_VHI = reshape(offset(3,:,1,:),[5 221]); % For frequencies from 100 to 320
% 8 April 2003: Choose frequencies from 190 to 290
%offset_VHI = reshape(offset(3,:,1,:),[5 101]);
offset_VHI = reshape(offset(3,:,1,:),[5 320]);
ds.avg_offset_VHI = sum(sum(offset_VHI,2)/size(offset_VHI,2))/(size(offset_VHI,1));

% For HQ channel
% offset_VHQ = reshape(offset(4,:,1,:),[5 221]); % For frequencies from 100 to 320
% 8 April 2003: Choose frequencies from 190 to 290
% offset_VHQ = reshape(offset(4,:,1,:),[5 101]);
offset_VHQ = reshape(offset(4,:,1,:),[5 320]);
ds.avg_offset_VHQ = sum(sum(offset_VHQ,2)/size(offset_VHQ,2))/(size(offset_VHQ,1));
%
% Take the repeated data values out in the signature file.
% First append one arbitrary number at the beginning of the file.
% Then find the difference of all data, if the same, the output is zero.
% Detect those zeros, and use only non zero data.
%
theta = tagdata(2:size(tagdata,1),2);
ds.theta_raw = theta;
nzeroindex=find((diff([1;theta]))~=0); % Pick out no repeated data
theta = theta(nzeroindex);
% Extract data
% Extract VI data for 320 frequencies and 401 azimuthal angles at range gate 3
% Only use 2.2 GHz bandwidth, corresponding from frequency 100 to frequency 320 (221 freqs).
% VIdata=reshape(data(receiver_pol,range_gate,pol_selection,100:320,2:402),221,401);
VIdata=reshape(data(receiver_pol,range_gate,pol_selection,1:320,2:402),320,401);
% VIdata after taking out repeated data due to the pedestal jitter because of
% small anlge step size.
VIdata=VIdata(:,nzeroindex);

% Extract VQ data for 320 frequencies and 401 azimuthal angles at range gate 3
% Only use 2.2 GHz bandwidth, corresponding from frequency 100 to frequency 320 (221 freqs).
% VQdata=reshape(data(receiver_pol+1,range_gate,pol_selection,100:320,2:402),221,401);
```

```
VQdata=reshape(data(receiver_pol+1,range_gate,pol_selection,1:320,2:402),320,401);
% VQdata after taking out repeated data due to the pedestal jitter because of
% small anlge step size.
VQdata=VQdata(:,nzeroindex);

if pol=='VV' | pol=='vv' | pol=='Vv' | pol== 'vV'
    I_data=VIdata-ds.avg_offset_VVI;
    Q_data=VQdata-ds.avg_offset_VVQ;
elseif pol=='VH' | pol=='vh' | pol=='Vh' | pol== 'vH'
    I_data=VIdata-ds.avg_offset_VHI;
    Q_data=VQdata-ds.avg_offset_VHQ;
elseif pol=='HH' | pol=='hh' | pol=='Hh' | pol== 'hH'
    I_data=VIdata-ds.avg_offset_VHI;
    Q_data=VQdata-ds.avg_offset_VHQ;
elseif pol=='HV' | pol=='hv' | pol=='Hv' | pol== 'hV'
    I_data=VIdata-ds.avg_offset_VVI;
    Q_data=VQdata-ds.avg_offset_VVQ;
end

% outdata=sum(I_data.^2+Q_data.^2)/221; % 3 April 2003: Choose frequencies from 100 to 320
% outdata=sum(I_data.^2+Q_data.^2)/101; % 8 April 2003: Choose frequencies from 190 to 290
outdata=sum(I_data.^2+Q_data.^2)/320; % 21 April 2003: Choose frequencies from 1 to 320

ds.theta=theta;
ds.VIdata=VIdata;
ds.VQdata=VQdata;
ds.I_data=I_data;
ds.Q_data=Q_data;
ds.IQ_data=I_data+j.*Q_data;

outdatadB=10*log10(outdata);
ds.outdata=outdata;
ds.outdatadB=outdatadB;
```

```
function Rxhatfss=estrxhatfs(x,subband)
% Filename: estrxhatfs.m (Generate estimated correlation matrix with forward)
% Author  : Canh Ly
% Date    : 21 March 1999
%
% Generate estimated correlation matrix with
% forward spatial smoothing for scanning antenna.
%
% Syntax: Rxhatfss=estrxhatfs(x,subband);
%
% where
%         x: input data matrix
%   Rxhatfss: estimated correlation matrix with forward
%              spatial smoothing.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This block used each snapshot to compute the submatrix and averages
% of all submatrices.
%
%P=subband;
%[N,K]=size(x);
%M=N-P+1; % Number of elements in each subarray
%sumsubRx=zeros(M);
%for k=1:K
%   M=N-P+1; % Reset new number of length index for each snapshot
%   xtemp=x(:,k);
%   for p=1:P
%      xs(:,p)=xtemp(p:M);
%      M=M+1;
%   end
%   sum=zeros(size(xs,1));
%   lenxs=size(xs,2);
%   for l=1:lenxs
%     R=xs(:,l)*xs(:,l)'; % xs(:,l)' is complex conj. transpose
     %Z= fliplr(flipud(conj(R)));
%     sum=sum+(R);
%   end
%   tempsum = sum/(P);
%   sumsubRx= sumsubRx+tempsum;
%   xs=[];xtemp=[];
%end
%Rxhatfss=sumsubRx/K;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This block uses the partition of sub-data matrix and then
% computes the subcorrelation matrix and then averages the
% subcorrelation matrices.
% K is number of snapshots, each column of the data matrix, x.
%
```

```
P=subband;
[N,K]=size(x);
M=N-P+1; % Number of elements in each subarray
Rxtemp=zeros(M,M);
flag=0;

for p=1:P
   Rxtemp=Rxtemp+x(p:M,:)*x(p:M,:)';
   M=M+1;
end
Rxhatfss=Rxtemp/(P*K);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function [eigval,eigvec]=eigvv(Rxhat)
% Filename: eigvv.m
% Author  : Canh Ly
% Date    : 6/15/95
%
% Purpose : Computes eigenvalues and eigenvectors
%
% Usage   : [eigval,eigvec]=eigvv(Rxhat);
%
[V,D]  =eig(Rxhat);
lambdai=diag(D);
[eigval,eigvec] = sorteig(lambdai,V);


function [signal,noise]=snspaces(eigval,eigvec,nsignals)
% Filename: snspaces.m
% Author  : Canh Ly
% Date    : March 6, 1997
%
% Purpose : Finds the number of signal eigenvalues and
%           theis indeces and splits noise and signal
%           space.
%
% Usage   : [signal,noise]=snspaces(eigval,eigvec,nsignals);
%
eigvalm    = eigval(1:nsignals);
eigvalindex=1:nsignals;
%
% Split signal and noise space vectors
%
ncoleigvec=size(eigvec,2);
noiseindex=1;
signalindex=1;
leneigvalindex=length(eigvalindex);
for j=1:ncoleigvec
    if(j == eigvalindex(signalindex))
      oldsignalindex=signalindex;
      signal(:,signalindex)=eigvec(:,j);
      signalindex=signalindex+1;
      if(signalindex>leneigvalindex)
         signalindex=oldsignalindex;
      end
    else
      noise(:,noiseindex)=eigvec(:,j);
      noiseindex=noiseindex+1;
    end
end
```

```
% Filename: plotsmuresult.m
% Author  : Canh Ly
% Date    : 25 March 2003
%
% Purpose: Plot the SMUSIC simulation result.
%
plot(az,pmusics0,'b');
% set(gca,'Xlim',[40 58])
axis([35 60 0 max(pmusics0)+5]);
title(plottitlestr);
grid on

% Filename: plottgtsetup.m
% Author  : Canh Ly
% Date    : 13 December 2001
%
% Purpose: Plot experiment target setup.
%
image(A);axis off;hold off

% Filename: plotmaveexpdata.m (Plot magnitude average experimental data)
% Author  : Canh Ly
% Date    : 20 June 2003
%
% Purpose: Plot magnitude average experimental data
%
plot(ds.theta-1.87,ds.outdatadB-min(ds.outdatadB),'r');
axis([35 60 0 max(pmusics0)+5]);
title(plottitlestr);
grid on

% Filename: plotpaveexpdata.m (Plot phase average experimental data)
% Author  : Canh Ly
% Date    : 20 June 2003
%
% Purpose: Plot phase average experimental data
%

avecomplex=(1/size(ds.IQ_data,1))*sum(ds.IQ_data);
plot(ds.theta-1.87,angle(avecomplex),'r');
grid on
```

```
%function plotmfreqdata(column)
% Filename: plotmfreqdata.m
% Author  : Canh Ly
% Date    : 20 June 2003
%
% Purpose: Plot magnitude data at a particular frequency.
%
%
hold off
column = get(gco,'value');

if column == size(ds.IQ_data,1)+1
    plot(ds.theta-1.87,ds.outdatadB-min(ds.outdatadB),'r');
else
    plot(ds.theta-1.87,10*log10(real(ds.IQ_data(column,:).^2+imag(ds.IQ_data(column,:).^2))))
end

xlabel('Azimuth angle (deg)')
ylabel('Amplitude')


%function plotpfreqdata(column)
% Filename: plotpfreqdata.m
% Author  : Canh Ly
% Date    : 20 June 2003
%
% Purpose: Plot phase across scanning angle at a particular frequency.
%
%
hold off
column = get(gco,'value');

if column == size(ds.IQ_data,1)+1
    avecomplex=(1/size(ds.IQ_data,1))*sum(ds.IQ_data);
    plot(ds.theta-1.87,angle(avecomplex),'r');
else
    plot(ds.theta-1.87,angle(ds.IQ_data(column,:)));
end

xlabel('Azimuth angle (deg)')
ylabel('Phase')
```

```
% Filename: plotantpat.m
% Author  : Canh Ly
% Date    : 24 March 2003
%
% Purpose: Plot 5 in antenna pattern. The data file in c:\smusic_MFRF\mfiles
%          was generated from ant=nor5ant; % This file output the ant structure
%
%          antraw=[ant.xant ant.yant];
%          antsmo=[ant.azscan ant.magscan];
%          save ant5inraw.dat antraw -ascii
%          save ant5insmo.dat antsmo -ascii
%
% 25 April 2003
load ant46in35GHz.dat
plot(ant46in35GHz(:,1)-3.77,ant46in35GHz(:,2),'k')
% End of 25 April 2003

grid on
axis([35 60 0 max(pmusics0)+5]);
title(plottitlestr);

% Shade the antenna beamwidth: Added on 21 Mar 2003
xantfill=[44.7 44.7 51.6 51.6];
yantfill=[0 max(pmusics0)+7.5 max(pmusics0)+7.5 0];
% yantfill=[0 25 25 0];
tcolor=[0.7 0.7 0.7];
p = fill(xantfill,yantfill,tcolor);
set(p,'FaceColor',tcolor,'EdgeColor','None','FaceAlpha',0.4);
```

```
%function plotpfreqant(column)
% Filename: plotpfreqant.m
% Author  : Canh Ly
% Date    : 21 July 2003
%
% Purpose: Plot phase across scanning angle at a particular frequency.
%
hold off
column = get(gco,'value');

if column == size(ant5.IQ_data_raw,2)+1
    load ant46in35GHz.dat
    plot(ant46in35GHz(:,1)-3.77,ant46in35GHz(:,2),'k')
    axis([35 60 0 max(pmusics0)+5]);
else
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Old procedure of correcting phase, correctantpha was created from smusic_mfrf_inter_Ver2.m
%
%     plot(ant5.theta,unwrap(correctantpha(:,column)),'b')
%     hold on;plot(ant5.theta,unwrap(angle(ant5.IQ_data_raw(:,column))),'r')
%     plot(ant5.theta,correctantpha(:,column),'b')
%     hold on;plot(ant5.theta,angle(ant5.IQ_data_raw(:,column)),'r')
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   New procedure of correcting phase
%
    plot(ant5.theta,unwrap(angle(ant5.IQ_data_raw_cor(:,column))),'b')
    hold on;plot(ant5.theta,unwrap(angle(ant5.IQ_data_raw_un(:,column))),'r')

    legendstr1=['correct @ freq ' num2str(column)];
    legendstr2=['uncorrect @ freq ' num2str(column)];
    legend(legendstr1,legendstr2)
    title('Phase Profile')
    grid
    hold off
end

xlabel('Azimuth angle (deg)')
ylabel('Phase (rad)')
```

Distribution

Admnstr
Defns Techl Info Ctr
ATTN DTIC-OCP (Electronic Copy)
8725 John J Kingman Rd Ste 0944
FT Belvoir VA 22060-6218

DARPA
ATTN IXO S Welby
3701 N Fairfax Dr
Arlington VA 22203-1714

Ofc of the Secy of Defns
ATTN ODDRE (R&AT)
The Pentagon
Washington DC 20301-3080

US Army TRADOC
Battle Lab Integration & Techl Dirctrt
ATTN ATCD-B
10 Whistler Lane
FT Monroe VA 23651-5850

US Army TRADOC
Battle Lab Integration & Techl Dirctrt
ATTN ATCD-B J A Klevecz
FT Monroe VA 23651-5850

Dir for MANPRINT Ofc of the
Deputy Chief of Staff for Prsnnl
ATTN J Hiller
The Pentagon Rm 2C733
Washington DC 20301-0300

US Military Acdmy
Mathematical Sci Ctr of Excellence
ATTN LTC T Rugenstein
Thayer Hall Rm 226C
West Point NY 10996-1786

SMC/GPA
2420 Vela Way Ste 1866
El Segundo CA 90245-4659

TECOM
ATTN AMSTE-CL
ATTN CSTE-DTC-CL
Aberdeen Proving Ground MD 21005-5057

US Army ARDEC
ATTN AMSTA-AR-TD
Bldg 1
Picatinny Arsenal NJ 07806-5000

US Army Avn & Mis Cmnd
ATTN AMSAM-RD M Schexneider
ATTN AMSAM-RD-MG-RF W Caraway
Redstone Arsenal AL 35898

US Army Avn & Mis Cmnd
ATTN AMSMI-RD W C McCorkle
Redstone Arsenal AL 35898-5240

US Army Info Sys Engrg Cmnd
ATTN AMSEL-IE-TD F Jenia
FT Huachuca AZ 85613-5300

US Army Natick RDEC
Acting Techl Dir
ATTN SBCN-TP P Brandler
Kansas Street Bldg78
Natick MA 01760-5056

US Army Simulation Train &
Instrmntn Cmnd
ATTN AMSTI-CG M Macedonia
12350 Research Parkway
Orlando FL 32826-3726

US Army TACOM
ATTN AMSTA-TR-R (Ms 263) J Soltesz
Warren MI 48397-5000

US Army Tank-Automtv Cmnd
ATTN AMSTA-TR-M J Lim
Warren MI 48397-5000

US Army Tank-Automtv Cmnd RDEC
ATTN AMSTA-TR J Chapin
Warren MI 48397-5000

Hicks & Assoc Inc
ATTN G Singley III
1710 Goodrich Dr Ste 1300
McLean VA 22102

Palisades Inst for Rsrch Svc Inc
ATTN E Carr
1745 Jefferson Davis Hwy Ste 500
Arlington VA 22202-3402

US Army Rsrch Lab
ATTN AMSRL-SE-RM S Stratton
Aberdeen Proving Ground MD 21005

US Army Rsrch Lab
ATTN AMSRL-WM-TA B Zoltoski
ATTN AMSRL-WM-TE A Niiler
ATTN AMSRL-WM-TE G Thompson
Aberdeen Proving Ground MD 21005-5066

Director
US Army Rsrch Lab
ATTN AMSRL-RO-D JCI Chang
ATTN AMSRL-RO-EN W D Bach
PO Box 12211
Research Triangle Park NC 27709

US Army Rsrch Lab
ATTN AMSRD-ARL-CI-IS-R Mail & Records Mgmt
ATTN AMSRD-ARL-CI-OK Techl Pub (2 copies)
ATTN AMSRD-ARL-CI-OK-TL Techl Lib (2 copies)
ATTN AMSRD-ARL-SE-R B Wallace
ATTN AMSRD-ARL-SE-RM C Ly (5 copies)
ATTN AMSRD-ARL-SE-RM D W Vance
ATTN AMSRD-ARL-SE-RM E Burke
ATTN AMSRD-ARL-SE-RM G Goldman
ATTN AMSRD-ARL-SE-RM J Nemarich
ATTN AMSRD-ARL-SE-RM K Tom
ATTN AMSRD-ARL-SE-RM R Harris
ATTN AMSRD-ARL-SE-RM R Wellman
Adelphi MD 20783-1197